PHILIPS

Data handbook

Electronic components and materials

Integrated circuits

Book IC14N 1985
New series

Microprocessors, microcontrollers and peripherals

Bipolar, MOS

MICROPROCESSORS, MICROCONTROLLERS AND PERIPHERALS

DATA HANDBOOK SYSTEM

Our Data Handbook System comprises more than 60 books with specifications on electronic components, subassemblies and materials. It is made up of four series of handbooks:

ELECTRON TUBES	BLUE
SEMICONDUCTORS	RED
INTEGRATED CIRCUITS	PURPLE
COMPONENTS AND MATERIALS	GREEN

The contents of each series are listed on pages iv to viii.

The data handbooks contain all pertinent data available at the time of publication, and each is revised and reissued periodically.

When ratings or specifications differ from those published in the preceding edition they are indicated with arrows in the page margin. Where application information is given it is advisory and does not form part of the product specification.

Condensed data on the preferred products of Philips Electronic Components and Materials Division is given in our Preferred Type Range catalogue (issued annually).

Information on current Data Handbooks and on how to obtain a subscription for future issues is available from any of the Organizations listed on the back cover.
Product specialists are at your service and enquiries will be answered promptly.

ELECTRON TUBES (BLUE SERIES)

The blue series of data handbooks comprises:

T1 **Tubes for r.f. heating**

T2a **Transmitting tubes for communications, glass types**

T2b **Transmitting tubes for communications, ceramic types**

T3 **Klystrons**

T4 **Magnetrons for microwave heating**

T5 **Cathode-ray tubes**
Instrument tubes, monitor and display tubes, C.R. tubes for special applications

T6 **Geiger-Müller tubes**

T7 **Gas-filled tubes** (will not be reprinted)

T8 **Colour display systems**
Colour TV picture tubes, colour data graphic display tube assemblies, deflection units

T9 **Photo and electron multipliers**

T10 **Plumbicon camera tubes and accessories**

T11 **Microwave semiconductors and components**

T12 **Vidicon and Newvicon camera tubes**

T13 **Image intensifiers**

T14 **Infrared detectors**

T15 **Dry reed switches**

Data collations on these subjects are available now.
Data Handbooks will be published in 1985.

T16 **Monochrome tubes and deflection units**
Black and white TV picture tubes, monochrome data graphic display tubes, deflection units

SEMICONDUCTORS (RED SERIES)

The red series of data handbooks comprises:

S1 **Diodes**
Small-signal germanium diodes, small-signal silicon diodes, voltage regulator diodes (< 1,5 W), voltage reference diodes, tuner diodes, rectifier diodes

S2a **Power diodes**

S2b **Thyristors and triacs**

S3 **Small-signal transistors**

S4a **Low-frequency power transistors and hybrid modules**

S4b **High-voltage and switching power transistors**

S5 **Field-effect transistors**

S6 **R.F. power transistors and modules**

S7 **Surface mounted semiconductors**

S8 **Devices for optoelectronics**
Photosensitive diodes and transistors, light-emitting diodes, displays, photocouplers, infrared sensitive devices, photoconductive devices.

S9 **Power MOS transistors**

S10 **Wideband transistors and wideband hybrid IC modules**

S11 **Microwave transistors**

S12 **Surface acoustic wave devices**

INTEGRATED CIRCUITS (PURPLE SERIES)

The purple series of data handbooks comprises:

EXISTING SERIES		Superseded by:
IC1	**Bipolar ICs for radio and audio equipment**	IC01N
IC2	**Bipolar ICs for video equipment**	IC02Na and IC02Nb
IC3	**ICs for digital systems in radio, audio and video equipment**	IC01N, IC02Na and IC02Nb
IC4	**Digital integrated circuits** CMOS HE4000B family	
IC5	**Digital integrated circuits – ECL** ECL10 000 (GX family), ECL100 000 (HX family), dedicated designs	IC08N
IC6	**Professional analogue integrated circuits**	
IC7	**Signetics bipolar memories**	
IC8	**Signetics analogue circuits**	IC11N
IC9	**Signetics TTL logic**	IC09N and IC15N
IC10	**Signetics Integrated Fuse Logic (IFL)**	IC13N
IC11	**Microprocessors, microcomputers and peripheral circuitry**	IC14N

NEW SERIES

IC01N	**Radio, audio and associated systems** Bipolar, MOS	(published 1985)
IC02Na	**Video and associated systems** Bipolar, MOS Types MAB8031AH to TDA1524A	(published 1985)
IC02Nb	**Video and associated systems** Bipolar, MOS Types TDA2501 to TEA1002	(published 1985)
IC03N	**Integrated circuits for telephony**	(published 1985)
IC04N	**HE4000B logic family** CMOS	
IC05N	**HE4000B logic family — uncased ICs** CMOS	(published 1984)
IC06N	**High-speed CMOS; PC54/74HC/HCT/HCU** Logic family	(published 1985)
IC07N	**High-speed CMOS; PC54/74HC/HCT/HCU — uncased ICs** Logic family	
IC08N	**ECL 10K and 100K logic families**	(published 1984)
IC09N	**TTL logic series**	(published 1984)
IC10N	**Memories** MOS, TTL, ECL	
IC11N	**Linear LSI**	(published 1985)
IC12N	**Semi-custom gate arrays & cell libraries** ISL, ECL, CMOS	
IC13N	**Semi-custom** Integrated Fuse Logic	(published 1985)
IC14N	**Microprocessors, microcontrollers & peripherals** Bipolar, MOS	(published 1985)
IC15N	**FAST TTL logic series**	(published 1984)

Note

Books available in the new series are shown with their date of publication.

COMPONENTS AND MATERIALS (GREEN SERIES)

The green series of data handbooks comprises:

C1 **Programmable controller modules**
PLC modules, PC20 modules

C2 **Television tuners, coaxial aerial input assemblies, surface acoustic wave filters**

C3 **Loudspeakers**

C4 **Ferroxcube potcores, square cores and cross cores**

C5 **Ferroxcube for power, audio/video and accelerators**

C6 **Synchronous motors and gearboxes**

C7 **Variable capacitors**

C8 **Variable mains transformers**

C9 **Piezoelectric quartz devices**

C10 **Connectors**

C11 **Non-linear resistors**
Voltage dependent resistors (VDR), light dependent resistors (LDR), negative temperature coefficient thermistors (NTC), positive temperature coefficient thermistors (PTC)

C12 **Potentiometers, encoders and switches**

C13 **Fixed resistors**

C14 **Electrolytic and solid capacitors**

C15 **Ceramic capacitors**

C16 **Permanent magnet materials**

C17 **Stepping motors and associated electronics**

C18 **Direct current motors**

C19 **Piezoelectric ceramics**

C20 **Wire-wound components for TVs and monitors**

C21* **Assemblies for industrial use**
HNIL FZ/30 series, NORbits 60-, 61-, 90-series, input devices

C22 **Film capacitors**

* Will be issued in 1985.

SELECTION GUIDE

FUNCTIONAL INDEX

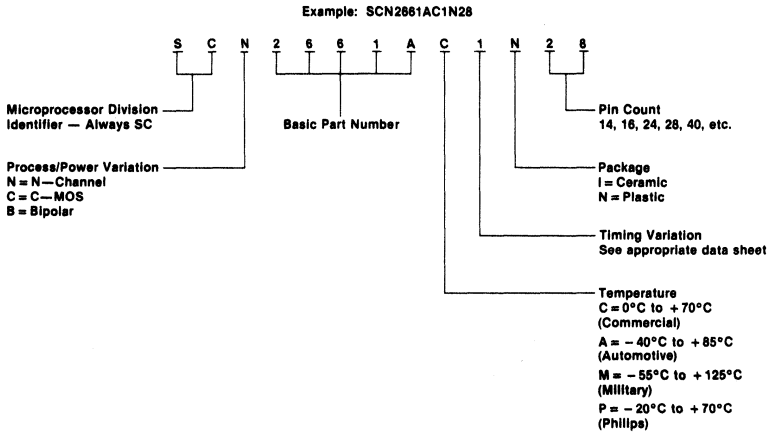
NUMERICAL INDEX

ORDERING INFORMATION

TYPE DESIGNATION

PRODUCT STATUS DEFINITIONS

PART NUMBERING SYSTEM



BIPOLAR LSI ORDERING INFORMATION

3X300 I/O Peripheral Components

Various 8X300/8X305 MicroController I/O parts and bus expanders can be ordered with an address preprogrammed at the factory or unprogrammed to permit feild address assignment. Addresses in range indicated as STOCK in the table below may be ordered in any quantity. Addresses outside of the STOCK range but indicated as AVAILABLE require a minimum order of 250 pieces per line item per part type. To order, use the part number indicated in the table, substituting the desired address of xx or xxx when ordering preprogrammed parts.

PART NUMBER	ADDRESSES		ORDER NUMBER	
	AVAILABLE	STOCK	UNPROGRAMMED	PREPROGRAMMED
N8T32F	000-255	None	N8T32F	N8T32F-xxx
N8T32N	000-255	000-015	N8T32N	N8T32N-xxx
N8T36F	000-255	None	N8T36F	N8T36F-xxx
N8T36N	000-255	000-015	N8T36N	N8T36N-xxx
N8X372N	000-255	000-015	N8X372N	N8X372N-xxx
N8X374N	000-255	000-015	N8X374N	N8X374-xxx
N8X376N	000-255	000-015	N8X376N	N8X376N-xxx
N8X382N	000-255	000-015	N8X382N	N8X382N-xxx

MCCAP 8X300/8X305 CROSSASSEMBLER PROGRAM

MCCAP, the crossassembler program for the 8X300 and 8X305 Micro-Controllers, is supplied as a 9-track magnetic tape containing FORTRAN IV source code for the crossassembler program. For compatibility with various computer systems, the tape is available in various combinations of density and data encoding. To order, use the following part numbers.

NUMBER	DENSITY	ENCODING
8X300 AS1-1 SS	800	ASCII
8X300 AS1-2 SS	800	EBCDIC
8X300 AS1-3 SS	1600	ASCII
8X300 AS1-4 SS	1600	EBCDIC
8X300 AS2SS	SINGLE/ DOUBLE	FLOPPY DISK

PRO ELECTRON TYPE DESIGNATION CODE FOR INTEGRATED CIRCUITS

This type nomenclature applies to semiconductor monolithic, semiconductor multi-chip, thin-film, thick-film and hybrid integrated circuits.

A basic number consists of:

THREE LETTERS FOLLOWED BY A SERIAL NUMBER

FIRST AND SECOND LETTER

1. DIGITAL FAMILY CIRCUITS

   The FIRST TWO LETTERS identify the FAMILY (see note 1).

2. SOLITARY CIRCUITS

   The FIRST LETTER divides the solitary circuits into:

   S : Solitary digital circuits
   T : Analogue circuits
   U : Mixed analogue/digital circuits

   The SECOND LETTER is a serial letter without any further significance except 'H' which stands for hybrid circuits.

3. MICROPROCESSORS

   The FIRST TWO LETTERS identify microprocessors and correlated circuits as follows:

   MA : { Microcomputer / Central processing unit
   MB : Slice processor (see note 2)
   MD : Correlated memories
   ME : Other correlated circuits (interface, clock, peripheral controller, etc.)

4. CHARGE-TRANSFER DEVICES AND SWITCHED CAPACITORS

   The FIRST TWO LETTERS identify the following:

   NH : Hybrid circuits
   NL : Logic circuits
   NM : Memories
   NS : Analogue signal processing, using switched capacitors
   NT : Analogue signal processing, using CTDs
   NX : Imaging devices
   NY : Other correlated circuits

Notes

1. A logic family is an assembly of digital circuits designed to be interconnected and defined by its basic electrical characteristics (such as: supply voltage, power consumption, propagation delay, noise immunity).
2. By 'slice processor' is meant: a functional slice of microprocessor.

THIRD LETTER

It indicates the operating ambient temperature range.
The letters A to G give information about the temperature:

A : temperature range not specified
B : 0 to + 70 °C
C : −55 to + 125 °C
D : −25 to + 70 °C
E : −25 to + 85 °C
F : −40 to + 85 °C
G : −55 to + 85 °C

If a circuit is published for another temperature range, the letter indicating a narrower temperature range may be used or the letter 'A'.

Example: the range 0 to + 75 °C can be indicated by 'B' or 'A'.

SERIAL NUMBER

This may be either a 4-digit number assigned by Pro Electron, or the serial number (which may be a combination of figures and letters) of an existing company type designation of the manufacturer.

To the basic type number may be added:

A VERSION LETTER

Indicates a minor variant of the basic type or the package. Except for 'Z', which means customized wiring, the letter has no fixed meaning. The following letters are recommended for package variants:

C : for cylindrical
D : for ceramic DIL
F : for flat pack
L : for chip on tape
P : for plastic DIL
Q : for QIL
T : for miniature plastic (mini-pack)
U : for uncased chip

Alternatively a TWO LETTER SUFFIX may be used instead of a single package version letter, if the manufacturer (sponsor) wishes to give more information.

FIRST LETTER: General shape

C : Cylindrical
D : Dual-in-line (DIL)
E : Power DIL (with external heatsink)
F : Flat (leads on 2 sides)
G : Flat (leads on 4 sides)
K : Diamond (TO-3 family)
M : Multiple-in-line (except Dual-, Triple-, Quadruple-in-line)
Q : Quadruple-in-line (QIL)
R : Power QIL (with external heatsink)
S : Single-in-line
T : Triple-in-line

SECOND LETTER: Material

C : Metal-ceramic
G : Glass-ceramic (cerdip)
M : Metal
P : Plastic

A hyphen precedes the suffix to avoid confusion with a version letter.

DEFINITION OF TERMS

Data Sheet Identification	Product Status	Definition
Preview	Formative or In Design	This data sheet contains the design specifications for product development. Specifications may change in any manner without notice.
Advance Information	Sampling or Pre-Production	This data sheet contains advance information and specifications are subject to change without notice.
Preliminary	First Production	This data sheet contains preliminary data and supplementary data will be published at a later date. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.
No Identification Noted	Full Production	This data sheet contains final specifications. Signetics reserves the right to make changes at any time without notice in order to improve design and supply the best possible product.

RATING SYSTEMS

RATING SYSTEMS

The rating systems described are those recommended by the International Electrotechnical Commission (IEC) in its Publication 134.

DEFINITIONS OF TERMS USED

Electronic device. An electronic tube or valve, transistor or other semiconductor device.

Note
This definition excludes inductors, capacitors, resistors and similar components.

Characteristic. A characteristic is an inherent and measurable property of a device. Such a property may be electrical, mechanical, thermal, hydraulic, electro-magnetic, or nuclear, and can be expressed as a value for stated or recognized conditions. A characteristic may also be a set of related values, usually shown in graphical form.

Bogey electronic device. An electronic device whose characteristics have the published nominal values for the type. A bogey electronic device for any particular application can be obtained by considering only those characteristics which are directly related to the application.

Rating. A value which establishes either a limiting capability or a limiting condition for an electronic device. It is determined for specified values of environment and operation, and may be stated in any suitable terms.

Note
Limiting conditions may be either maxima or minima.

Rating system. The set of principles upon which ratings are established and which determine their interpretation.

Note
The rating system indicates the division of responsibility between the device manufacturer and the circuit designer, with the object of ensuring that the working conditions do not exceed the ratings.

ABSOLUTE MAXIMUM RATING SYSTEM

Absolute maximum ratings are limiting values of operating and environmental conditions applicable to any electronic device of a specified type as defined by its published data, which should not be exceeded under the worst probable conditions.

These values are chosen by the device manufacturer to provide acceptable serviceability of the device, taking no responsibility for equipment variations, environmental variations, and the effects of changes in operating conditions due to variations in the characteristics of the device under consideration and of all other electronic devices in the equipment.

The equipment manufacturer should design so that, initially and throughout life, no absolute maximum value for the intended service is exceeded with any device under the worst probable operating conditions with respect to supply voltage variation, equipment component variation, equipment control adjustment, load variations, signal variation, environmental conditions, and variations in characteristics of the device under consideration and of all other electronic devices in the equipment.

DESIGN MAXIMUM RATING SYSTEM

Design maximum ratings are limiting values of operating and environmental conditions applicable to a bogey electronic device of a specified type as defined by its published data, and should not be exceeded under the worst probable conditions.

These values are chosen by the device manufacturer to provide acceptable serviceability of the device, taking responsibility for the effects of changes in operating conditions due to variations in the characteristics of the electronic device under consideration.

The equipment manufacturer should design so that, initially and throughout life, no design maximum value for the intended service is exceeded with a bogey device under the worst probable operating conditions with respect to supply voltage variation, equipment component variation, variation in characteristics of all other devices in the equipment, equipment control adjustment, load variation, signal variation and environmental conditions.

DESIGN CENTRE RATING SYSTEM

Design centre ratings are limiting values of operating and environmental conditions applicable to a bogey electronic device of a specified type as defined by its published data, and should not be exceeded under normal conditions.

These values are chosen by the device manufacturer to provide acceptable serviceability of the device in average applications, taking responsibility for normal changes in operating conditions due to rated supply voltage variation, equipment component variation, equipment control adjustment, load variation, signal variation, environmental conditions, and variations in the characteristics of all electronic devices.

The equipment manufacturer should design so that, initially, no design centre value for the intended service is exceeded with a bogey electronic device in equipment operating at the stated normal supply voltage.

HANDLING MOS DEVICES

HANDLING MOS DEVICES

Though all our MOS integrated circuits incorporate protection against electrostatic discharges, they can nevertheless be damaged by accidental over-voltages. In storing and handling them, the following precautions are recommended.

Caution

Testing or handling and mounting call for special attention to personal safety. Personnel handling MOS devices should normally be connected to ground via a resistor.

Storage and transport

Store and transport the circuits in their original packing. Alternatively, use may be made of a conductive material or special IC carrier that either short-circuits all leads or insulates them from external contact.

Testing or handling

Work on a conductive surface (e.g. metal table top) when testing the circuits or transferring them from one carrier to another. Electrically connect the person doing the testing or handling to the conductive surface, for example by a metal bracelet and a conductive cord or chain. Connect all testing and handling equipment to the same surface.
Signals should not be applied to the inputs while the device power supply is off. All unused input leads should be connected to either the supply voltage or ground.

Mounting

Mount MOS integrated circuits on printed circuit boards *after* all other components have been mounted. Take care that the circuits themselves, metal parts of the board, mounting tools, and the person doing the mounting are kept at the same electric (ground) potential. If it is impossible to ground the printed-circuit board the person mounting the circuits should touch the board before bringing MOS circuits into contact with it.

Soldering

Soldering iron tips, including those of low-voltage irons, or soldering baths should also be kept at the same potential as the MOS circuits and the board.

Static charges

Dress personnel in clothing of non-electrostatic material (no wool, silk or synthetic fibres). After the MOS circuits have been mounted on the board proper handling precautions should still be observed. Until the sub-assemblies are inserted into a complete system in which the proper voltages are supplied, the board is no more than an extension of the leads of the devices mounted on the board. To prevent static charges from being transmitted through the board wiring to the device it is recommended that conductive clips or conductive tape be put on the circuit board terminals.

Transient voltages

To prevent permanent damage due to transient voltages, do not insert or remove MOS devices, or printed-circuit boards with MOS devices, from test sockets or systems with power on.

Voltage surges

Beware of voltage surges due to switching electrical equipment on or off, relays and d.c. lines.

8-BIT SINGLE-CHIP MICROCONTROLLERS

SINGLE-CHIP 8-BIT MICROCOMPUTER

DESCRIPTION

The MAB8021 is a single-chip 8-bit microcomputer that is fabricated using the N-MOS silicon gate process. It contains a 1K x 8 program memory, a 64 x 8 data memory, 21 I/O lines, and an 8-bit timer/event counter, in addition to on-board oscillator and clock circuits. For systems that require extra I/O capacity, the MAB8021 can be expanded using the 8243 or discrete logic.
This microcomputer is designed to be an efficient controller as well as an arithmetic computer. The MAB8021 has bit handling capability as well as facilities for both binary and BCD arithmetic. Efficient use of program memory results from an instruction set (see Table 4) consisting mostly of single byte instructions and no instructions over two bytes in length.

FEATURES

- 8-bit CPU, ROM, RAM, I/O in a single 28-lead package
- 1K x 8 ROM, 64 x 8 RAM, 21 I/O lines
- Internal timer/event counter
- Clock generated with single inductor or crystal
- Single 5 V supply (range: + 4,5 V to + 6,5 V)
- 8,38 μs cycle time; all instructions 1 or 2 cycles
- Instructions: MAB8048 subset
- Zero-cross detection capability
- Easily expandable I/O

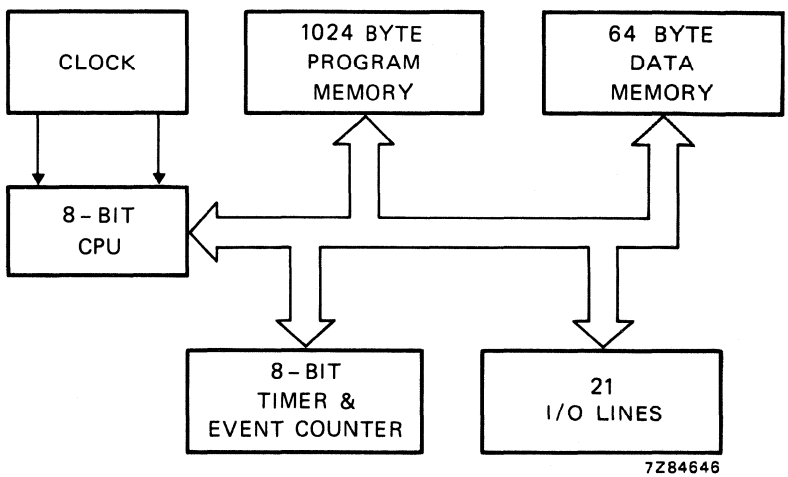


Fig. 1 Block diagram.

PACKAGE OUTLINES

MAB/F8021P: 28-lead DIL; plastic (SOT-117).

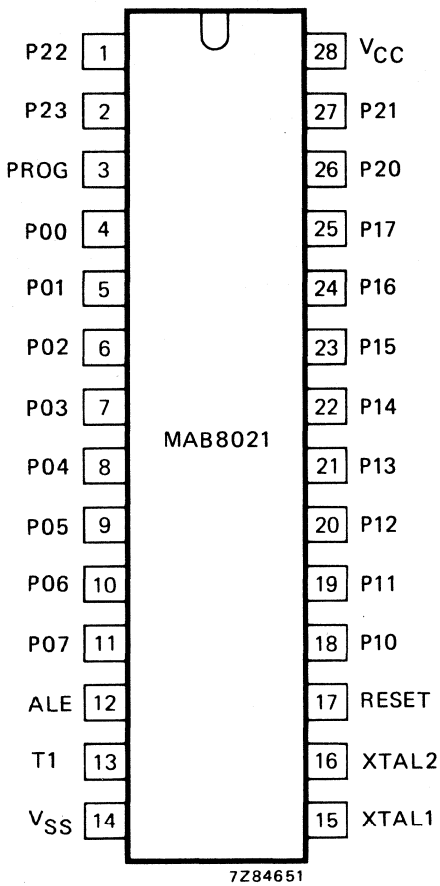


Fig. 2 Pinning diagram.

PIN DESIGNATION

designation	pin no.	function
V_{CC}	28	Power supply: + 5,5 V.
V_{SS}	14	Ground.
PROG	3	Output strobe: for 8243 I/O expander.
P00-P07	4-11	Port 0: 8-bit quasi-bidirectional port.
P10-P17	18-25	Port 1: 8-bit quasi-bidirectional port.
P20-P23	26,27,1,2	Port 2: 4-bit quasi-bidirectional port. P20-P23 also serve as a 4-bit I/O expander bus for the 8243.
T1	13	T1: input pin testable using the JT1 and JNT1 instructions. Can be designated the timer/event counter input, using the STRT CNT instructions. Also allows zero-crossover sensing of slowly moving a.c. inputs.
RESET	17	Reset: input, used to initialize the processor by clearing status flip-flops and setting program counter to zero.
ALE	12	Address latch enable: signal occurring once every 30 input clocks, used as an output clock.
XTAL1	15	Timing control element: one side of crystal or inductor input for internal oscillator. Also the input for an external source (not TTL compatible).
XTAL2	16	Timing control element: other side.

FUNCTIONAL DESCRIPTION

The following is a functional description of the MAB8021.

Program memory

The program memory consists of a 1024 byte mask-programmed ROM. No external expansion capability is provided. The first instruction must be at location 0.

Page organization

The program memory is organized as four pages of 256 bytes each. Only the unconditional branch instruction (JMP) can cause jump over page boundaries. The CALL instruction can transfer control to a subroutine on any page. Likewise, the table data reference instruction (MOVP A,@,A) can only access data located on the same page as the instruction.

Data memory

The 64 byte dynamic RAM is organized as shown in Fig. 3. Locations (registers) 0 to 7 are directly addressable by using several instructions. All locations are indirectly addressable by using registers R0 or R1.

Address stack

Locations 8 to 23 are used as the address stack. The address stack enables the processor to keep track of the return addresses generated from CALL instructions. A 3-bit stack pointer (SP) supplies the address of the locations to be loaded with the next return address generated. The SP to this push-down stack is incremented by one after a return address is stored, and decremented by one before an address is fetched during a RET (return instruction). A total of 8 levels of nesting is possible. The SP is initialized to location 8 upon RESET.
Since each address is 10-bits long, two bytes must be used to store a single address. The SP is incremented and decremented by one, but each increment or decrement moves the address pointed to, by two. Therefore, only even numbered addresses are pointed to. If a particular application does not require 8 levels of nesting, the unused portion of the stack may be used as any other indirectly addressable scratchpad location.

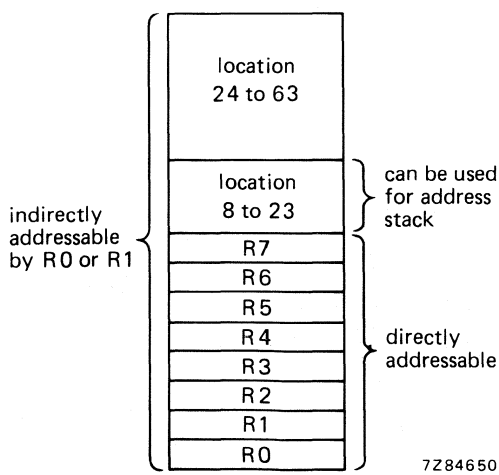


Fig. 3 Internal RAM memory map.

FUNCTIONAL DESCRIPTION (continued)

Program variable storage

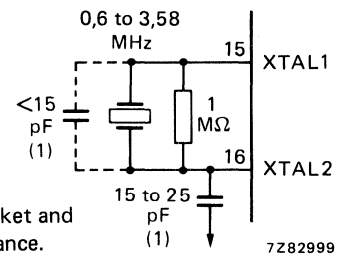
Locations 24 to 63 may be used for storage of program variables or data. In addition, any portion of the stack area (locations 8 to 23) not used for return address storage may also be used for variable storage.

Oscillator and clock

The MAB8021 contains its own internal oscillator and clock driver. The frequency is determined by a single crystal, or inductor. All internal time slots are derived from the external element, and all outputs are a function of the oscillator frequency. The particular control element is connected between pins 15 (XTAL1) and 16 (XTAL2).

An instruction cycle consists of 10 states, and each state is a time slot of 3 oscillator periods. Therefore, to obtain a 10 μs instruction cycle, a 3 MHz crystal should be used.

The MAB8021 utilizes dynamic RAM and certain other dynamic logic. Due to the clocking required with dynamic circuits, the oscillator frequency must be equal to or greater than 600 kHz, or improper operation may occur.



(1) Including socket and stray capacitance.

Counter/timer

The MAB8021 has an internal counter that can, under program control, count either external events (T1, pin 13) or instruction cycles. The instructions that control the counter and the functions they perform are summarized in Table 1.

The counter is an 8-bit binary up-counter. When used as a timer, the input to the counter is the overflow of a 5-bit (÷ 32) prescaler. The input of the prescaler is a signal that occurs each instruction cycle (30 clock periods).

When used as a counter, HIGH-to-LOW transitions on the T1 input are counted. The maximum frequency that can be counted is one third of the instruction cycle frequency (33,3 kHz for a 3 MHz oscillator). The positive duration of the signal must be a minimum of one tenth of the period of the instruction cycle.

When the 8-bit counter overflows, a flag is set. The flag can be tested using the JTF (jump if timer flag = 1) instruction, which resets the flag.

Table 1 Counter/timer control

function	used together counts instruction cycles		counter
	prescaler	timer	
CLEAR	STRT T	MOV T,A@(A) = 0	MOV T,A@(A) = 0
PRESET	–	MOV T,A	MOV T,A
START	STRT T	STRT T	STRT CNT
STOP	STOP TCNT	STOP TCNT or RESET	STOP TCNT or RESET
TEST	–	JTF	JTF
READ*	–	MOV A,T	MOV A,T

* READ does not disturb the counting process.

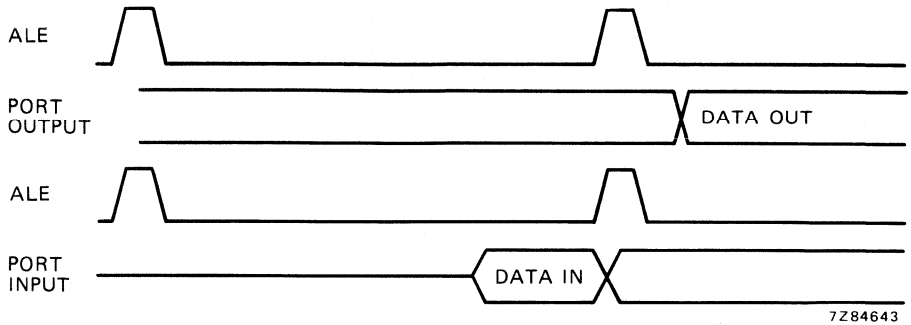


Fig. 4 Port 2 timing; normal operation.

Input/output

The 21 I/O pins are arranged as two 8-bit quasi-bidirectional ports, one 4-bit quasi-bidirectional port, and the T1 input. The 20 port pins can be individually assigned, under program control, to be either inputs or outputs (at a given time) or both (at different times).

Port operation

A simplified representation of the port circuitry is shown in Fig. 5. Output data written to a port is latched and remains unchanged until rewritten. Writing a '1' to the ports turns on the large pull-up device for less than the instruction time, which reduces the rise time of the signal. The smaller pull-up device then maintains the '1' level. The on-resistance of the small pull-up is approximately 30 kΩ. This means that, when used as an input pin, the external circuitry must sink only 0,2 mA at V_{IL}, so the MAB8021 recognizes a '0'. Thus, writing a '1' to a port-pin, it is enabled to be used either as an input pin or as a true HIGH level, latched, output pin. Writing a '0' to a port, a large pull-down device is turned on, which is capable of sinking 1,6 mA into V_{SS}. The preceding applies to ports 1 and 2.

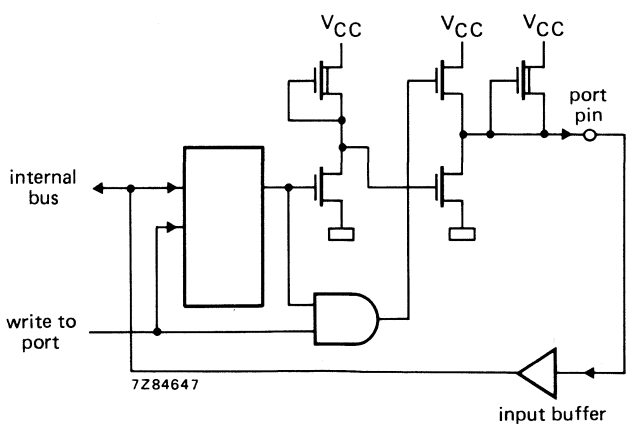


Fig. 5 Quasi-bidirectional port.

FUNCTIONAL DESCRIPTION (continued)

Port operation (continued)

Port 0 (P00 to P07) does not have the large pull-up device. In addition, by mask option, the small pull-up device (on any pin) can be deleted, thus providing a true open drain output.
The instructions ANL and ORL can be used to mask the ports, line-by-line, so that any combination of inputs and outputs can be configured on any port.

High-current outputs

Two pins (P10 and P11) are provided, which can sink larger currents (7 mA each, at V_{SS} + 2,5 V), when required. These pins may be used in parallel for 14 mA drive if the output logic states are always the same.

T1 input

The T1 input can be used for the following functions:
- event counter (external input)
- test input for branch instructions
- zero voltage cross-over detection.

The operation of T1 as an input to the event counter is described under the heading "Counter/timer".
When used as a test input, the JT1 and JNT1 instructions test for '1' and '0' levels respectively.
The T1 pin can also be used to detect the zero-crossing of slowly moving a.c. signals (50 Hz). The self-biasing circuit shown in Fig. 6 permits the T1 input to detect when the input voltage crosses zero within ± 135 mV, when the voltage is coupled through a 0,2 μF capacitor. The maximum input voltage is 3 V peak-to-peak. The zero-cross detection is especially useful in thyristor control of 50 Hz power equipment and in developing time-of-day and other timing routines. A pull-up resistor can be provided as a ROM mask option. This is useful when the input is from a switch or a standard TTL output.

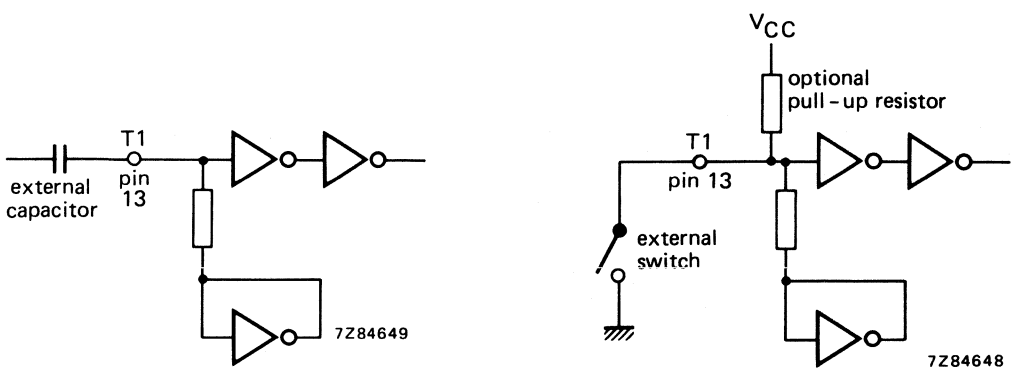


Fig. 6 T1 pin options for (a) zero-cross applications and with (b) optional pull-up resistor.

Expanded I/O

The MAB8021 can be used with the 8243 I/O expander chip, which provides additional I/O capability. The 8243 has 4 directly addressable 4-bit ports. It connects the PROG pin, which provides the clock, and pins P20 to P23, which provide address and data. These ports can be written with a MOVD P,A, ANLD P,A, and ORLD P,A for ports 4 to 7. Reading is via MOVD A,P instruction. The previous data on P20 to P23, before an output expander instruction, is lost. Therefore, when using an output expander, P20 to P23 are not useful for general input/output operation. This circuit configuration is shown in Fig. 7 and the timing diagram in Fig. 8.
The MAB8021 can also use standard low-cost TTL circuits, to expand the number of I/O lines.

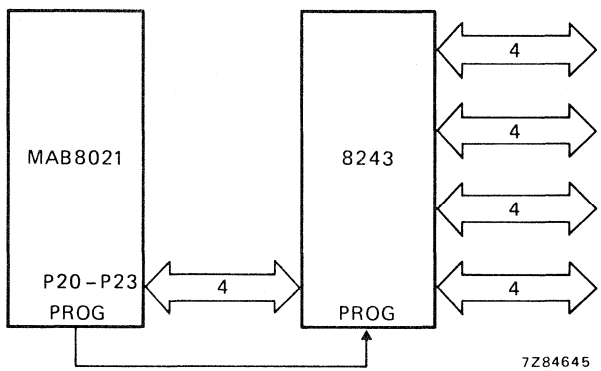


Fig. 7 Expander interface.

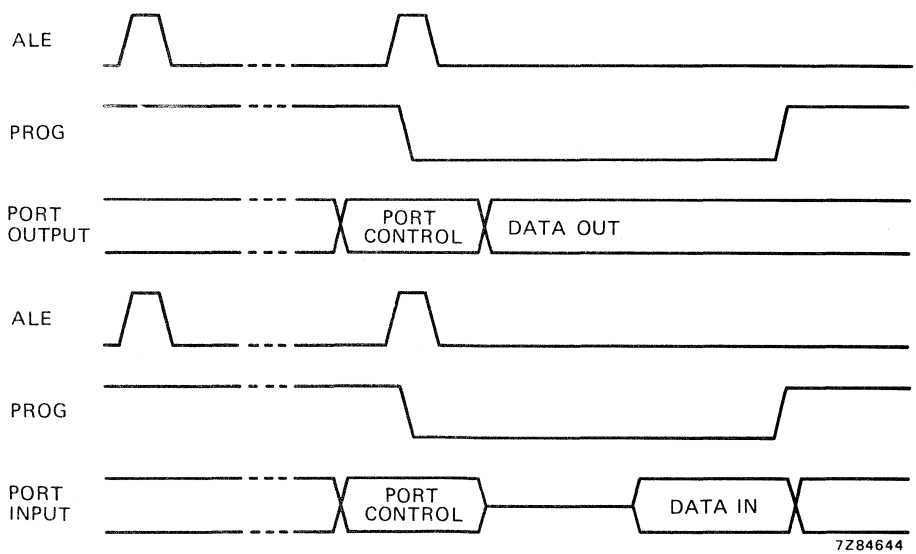


Fig. 8 Port 2 timing; expander operation.

FUNCTIONAL DESCRIPTION (continued)

Central Processing Unit

The MAB8021 has arithmetic, logical and conditional branching capabilities. The DA A, SWAP A, and XCHD instructions simplify BCD arithmetic and handling of nibbles. The MOVP A,@A instruction permits efficient table look-up from the current ROM page.
The conditional branch logic within the processor enables several conditions, internal and external to the processor, to be tested by the user's program. In addition, the DJNZ instruction decrements a designated register and branches if the contents are not zero. This instruction is useful for looping control.

Testing and debug

To facilitate testing and debug, certain test modes may be activated in the MAB8021 by raising combinations of RESET, T1 and PROG to 15 V. The internal ROM is dumped out sequentially for verification. External memory operation is used for CPU check-out.

RESET	PROG	T1	case	function
5 V	X	X		power-on clear
0 V	X	X		normal operation
15 V	15 V	⎍	mode 1b	On every T1 rising edge the program counter increments, dumps internal ROM to Port 0.
0 V	15 V	X	mode 2	IC will operate from external memory (one page) via Port 0. ALE strobes address out, memory in.
15 V	X	X	mode 3	IC accepts op-codes into Port 1; allows Port 0 and 8243 testing.

X = normal mode (between 0 V and V_{CC}).

Differences between the MAB8021 and the MAB8048

Although the MAB8021 is basically an electrical and functional subset of the MAB8048, there are some differences:

1. Pin out — As the MAB8021 is a 28-lead DIL, some form of adapter must be used to interface the MAB8021 socket to ICE-49. An emulation board (EM-1) has been designed to perform this function. The EM-1 also accounts for the increased flexibility of some MAB8021 I/O lines.
2. Instruction time — The MAB8021 instruction cycle is 30 clock cycles long, the MAB8048 instruction cycle is 15 clock cycles long. Where exact timing is important, the MAB8048 breadboard part should be operated at half the MAB8021 clock rate.
3. Test 1 — To facilitate developing time-of-day routine, and for thyristor control, the T1 pin (without the pull-up resistor option) will detect zero-crossing of a capacitively coupled a.c. input.
4. Quasi-bidirectional Ports — All MAB8021 ports are quasi-bidirectional to facilitate stand-alone use. Port 0 has open drain outputs and by mask option it may or may not have pull-up resistors.
5. Oscillator — The MAB8021 has an on-chip oscillator that is optimized for the single inductor mode. External connection will differ from the MAB8048.
6. Timer/counter — 1. If prescaler overflow occurs during a 'STRT T' or 'STOP TCNT' instruction, the MAB8048 will increment the timer, while the MAB8021 will not.
   2. The MAB8021 sets the timer flag in the same cycle as the overflow. The MAB8048 waits one cycle. Therefore, the MAB8021 can do a JTF instruction one cycle earlier (prescaler = '0') than the MAB8048 (prescaler = '1').
   3. The MAB8048 doesn't increment its timer in the second cycle of a 2-cycle instruction; the MAB8021 does.
7. High-current outputs — Very high current drive is desirable for minimizing external parts required to do high-power control. P10 and P11 have been designated high drive outputs capable of sinking 7 mA at V_{SS} + 2,5 V. (For clarity, this is a 7 mA to V_{SS} with a 2,5 V drop across the buffer.) These pins may, of course, be used in parallel for 14 mA drive if the output logic states are always the same.
8. Reset — Reset has been modified on the MAB8021. On the MAB8021, RESET is active HIGH; on the MAB8048, active LOW. Also, the MAB8021 does not reset the timer flag, while the MAB8048 does.
9. Instruction set — The following instructions (see Table 2), which are found in the MAB8048, have been deleted from the MAB8021 instruction set.

Table 2 Instruction set differences

DATA MOVES	REGISTERS	BRANCH	SUBROUTINE	CONTROL	INPUT/OUTPUT
MOV A,PSW	DEC R	JT0 addr	RETR	EN I	ANL P,#data
MOV PSW,A	FLAGS	JNT0 addr	TIMER	DIS I	ORL P,#data
MOVX A,@R		JF0 addr		SEL RB0	INS A,BUS*
MOVX @R,A	CLR F0	JF1 addr	EN TCNTI	SEL RB1	OUTL BUS,A*
MOVP3 A,@A	CPL F0	JNI addr	DIS TCNTI	SEL MB0	ANL BUS,#data
	CLR F1	JBb addr		SEL MB1	ORL BUS,#data
	CPL F1			ENT0 CLK	

* These instructions have been replaced in the MAB8021 by INA,P0 and OUTL P0,A respectively. OUTL P0,A has op-code of MOVX @R0,A.

Table 3 Instruction set

	mnemonic		description	bytes	cycles
ACCUMULATOR	ADD	A,R	Add register to A	1	1
	ADD	A,@R	Add data memory to A	1	1
	ADD	A,#data	Add immediate to A	2	2
	ADDC	A,R	Add with carry	1	1
	ADDC	A,@R	Add with carry	1	1
	ADDC	A,#data	Add with carry	2	2
	ANL	A,R	AND register to A	1	1
	ANL	A,@R	AND data memory to A	1	1
	ANL	A,#data	AND immediate to A	2	2
	ORL	A,R	OR register to A	1	1
	ORL	A,@R	OR data memory to A	1	1
	ORL	A,#data	OR immediate to A	2	2
	XRL	A,R	Exclusive OR register to A	1	1
	XRL	A,@R	Exclusive OR data memory to A	1	1
	XRL	A,#data	Exclusive OR immediate to A	2	2
	INC	A	Increment A	1	1
	DEC	A	Decrement A	1	1
	CLR	A	Clear A	1	1
	CPL	A	Complement A	1	1
	DA	A	Decimal adjust A	1	1
	SWAP	A	Swap nibbles of A	1	1
	RL	A	Rotate A left	1	1
	RLC	A	Rotate A left through carry	1	1
	RR	A	Rotate A right	1	1
	RRC	A	Rotate A right through carry	1	1
INPUT/OUTPUT	IN	A,P	Input port to A	1	2
	OUTL	P,A	Output A to port	1	2
	MOVD	A,P	Input expander port to A	1	2
	MOVD	P,A	Output A to expander port	1	2
	ANLD	P,A	AND A to expander port	1	2
	ORLD	P,A	OR A to expander port	1	2

DEVELOPMENT DATA

	mnemonic		description	bytes	cycles
REG.	INC	R	Increment register	1	1
	INC	@R	Increment data memory	1	1
BRANCH	JMP	addr	Jump unconditional	2	2
	JMPP	@A	Jump in page indirect	1	2
	DJNZ	R, addr	Decrement register and jump on R not zero	2	2
	JC	addr	Jump on carry = 1	2	2
	JNC	addr	Jump on carry = 0	2	2
	JZ	addr	Jump on A zero	2	2
	JNZ	addr	Jump on A not zero	2	2
	JT1	addr	Jump on T1 = 1	2	2
	JNT1	addr	Jump on T1 = 0	2	2
	JTF	addr	Jump on timer flag	2	2
SUBR.	CALL		Jump to subroutine	2	2
	RET		Return	1	2
FLAGS	CLR	C	Clear Carry	1	1
	CPL	C	Complement Carry	1	1
DATA MOVES	MOV	A,R	Move register to A	1	1
	MOV	A,@R	Move data memory to A	1	1
	MOV	A, # data	Move immediate to A	2	2
	MOV	R,A	Move A to register	1	1
	MOV	@R,A	Move A to data memory	1	1
	MOV	R, # data	Move immediate to register	2	2
	MOV	@R, # data	Move immediate to data memory	2	2
	XCH	A,R	Exchange A and register	1	1
	XCH	A, @R	Exchange A and data memory	1	1
	XCHD	A,@R	Exchange lower nibble of A and data memory	1	1
	MOVP	A,@A	Move to A from current page	1	2
TIMER/ COUNTER	MOV	A,T	Read timer/counter	1	1
	MOV	T,A	Load timer/counter	1	1
	STRT	T	Start timer	1	1
	STRT	CNT	Start counter	1	1
	STOP	TCNT	Stop timer/counter	1	1
	NOP		No operation	1	1

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Supply voltage with respect to V_{SS}	V_{CC}	−0,5 to + 7	V
Voltage on any input or output	V_I, V_O	−0,5 to + 7	V
D.C. current into any input or output	$\pm I_I$, $\pm I_O$	max. 10	mA
Total power dissipation	P_{tot}	max. 1	W
Storage temperature range	T_{stg}	−65 to + 150	°C
Operating ambient temperature range	T_{amb}	0 to + 70	°C

D.C. CHARACTERISTICS

V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; all voltages with respect to V_{SS}; unless otherwise specified

	symbol	min.	typ.	max.		conditions
Supply voltage	V_{CC}	4,5	5,5	6,5	V	
Supply current	I_{CC}	–	–	75	mA	
Input voltage LOW	V_{IL}	−0,5	–	0,8	V	
Input voltage HIGH all inputs except XTAL1,XTAL2,T1,RESET	V_{IH}	3,0	–	V_{CC}	V	V_{CC} = 5,5 + 1 V
Input voltage HIGH XTAL1,XTAL2,T1,RESET	V_{IH}	3,8	–	V_{CC}	V	V_{CC} = 5,5 + 1 V
Input voltage HIGH (10%) all inputs except XTAL1,XTAL2,T1,RESET	V_{IH}	2,0	–	V_{CC}	V	V_{CC} = 5,0 V ± 10%
Input voltage HIGH (10%) XTAL1,XTAL2,T1,RESET	V_{IH}	3,5	–	V_{CC}	V	V_{CC} = 5,0 V ± 10%
Output voltage LOW	V_{OL}	–	–	0,45	V	I_{OL} = 1,6 mA
Output voltage LOW P10, P11	V_{OL}	–	–	2,5	V	I_{OL} = 7 mA
Output voltage HIGH all outputs unless open drain	V_{OH}	2,4	–	–	V	$-I_{OH}$ = 50 μA
Output leakage current open drain option-port 0	$\pm I_{OL}$	–	–	10	μA	$V_{CC} \geqslant V_I \geqslant V_{SS}$ + 0,45 V

A.C. CHARACTERISTICS

V_{SS} = 0 V; V_{CC} = 5,5 V ± 1 V; T_{amb} = 0 to + 70 °C; unless otherwise specified

	symbol	min.	typ.	max.		
Cycle time	t_{cy}	8,38	–	50,0	μs	3,58 MHz crystal = 8,38 μs
T1 zero-cross characteristics						
Zero-cross detection input (T1) voltage; peak-to-peak value	$V_{zx(p-p)}$	....	....	....	V	a.c. coupled; C = 0,2 μF
Zero-cross accuracy 50 Hz sine-wave	A_{zx}	....	....	....	mV	
Zero-cross detection input frequency (T1)	f_{T1}	....	....	....	kHz	

DEVELOPMENT DATA

SINGLE-CHIP 8-BIT MICROCONTROLLER

DESCRIPTION

The MAB8051AH family of single-chip 8-bit microcontrollers is manufactured in an advanced 2 μ NMOS process. The family consists of the following members:

- MAB8031AH: ROM-less version of the MAB8051AH
- MAB8051AH: 4 K bytes mask-programmable ROM, 128 bytes RAM

Both types are available in 8, 10 and 12 MHz versions and 15 MHz for the MAB8031AH. In the following, the generic term "MAB8051AH" is used to refer to both family members.

The device provides hardware features, architectural enhancements and new instructions to function as a controller for applications requiring up to 64 K bytes of program memory and/or up to 64 K bytes of data storage.

The MAB8051AH contains a non-volatile 4 K x 8 read-only program memory (not ROM-less version); a volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/event counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O power for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and timing circuits. For systems that require extra capability, the MAB8051AH can be expanded using standard TTL compatible memories and logic.

The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of 255 instructions; 44% one-byte, 41% two-byte and 15% three-byte. With a 12 MHz crystal, 58% of the instructions are executed in 1 μs and 40% in 2 μs. Multiply and divide instructions require 4 μs. Multiply, divide, subtract and compare are among the many instructions added to the standard MAB8048H instruction set.
For further detailed information see Users Manual 'Single-chip microcomputer'.

Features

- 4 K x 8 ROM (8051AH only), 128 x 8 RAM
- Four 8-bit ports, 31 I/O lines
- Two 16-bit timer/event counters
- Full duplex serial port
- External memory expandable to 128 K
- Boolean processing
- 218 bit-addressable locations
- On-chip oscillator
- Five-source interrupt structure with two priority levels
- 58% of instructions executed in 1 μs; multiply and divide in 4 μs (at 12 MHz clock)
- Enhanced architecture with:
  - non-page-oreinted instructions
  - direct addressing
  - four 8-bit register banks
  - stack depth up to 128-bytes
  - multiply, divide, subtract and compare
- Available with extended temperature range −40 to + 85 °C (MAF8031/51AH)
  −40 to + 100 °C (MAF80A31/51AH)

PACKAGE OUTLINES

MAB8031/51AHP; MAF8031/51AHP; MAF80A31/51AHP: 40-lead DIL; plastic (SOT-129).
MAB8031/51AHWP; MAF8031/51AHWP; MAF80A31/51AHWP: 44-lead, plastic leaded-chip-carrier (PLCC); SOT-187A.

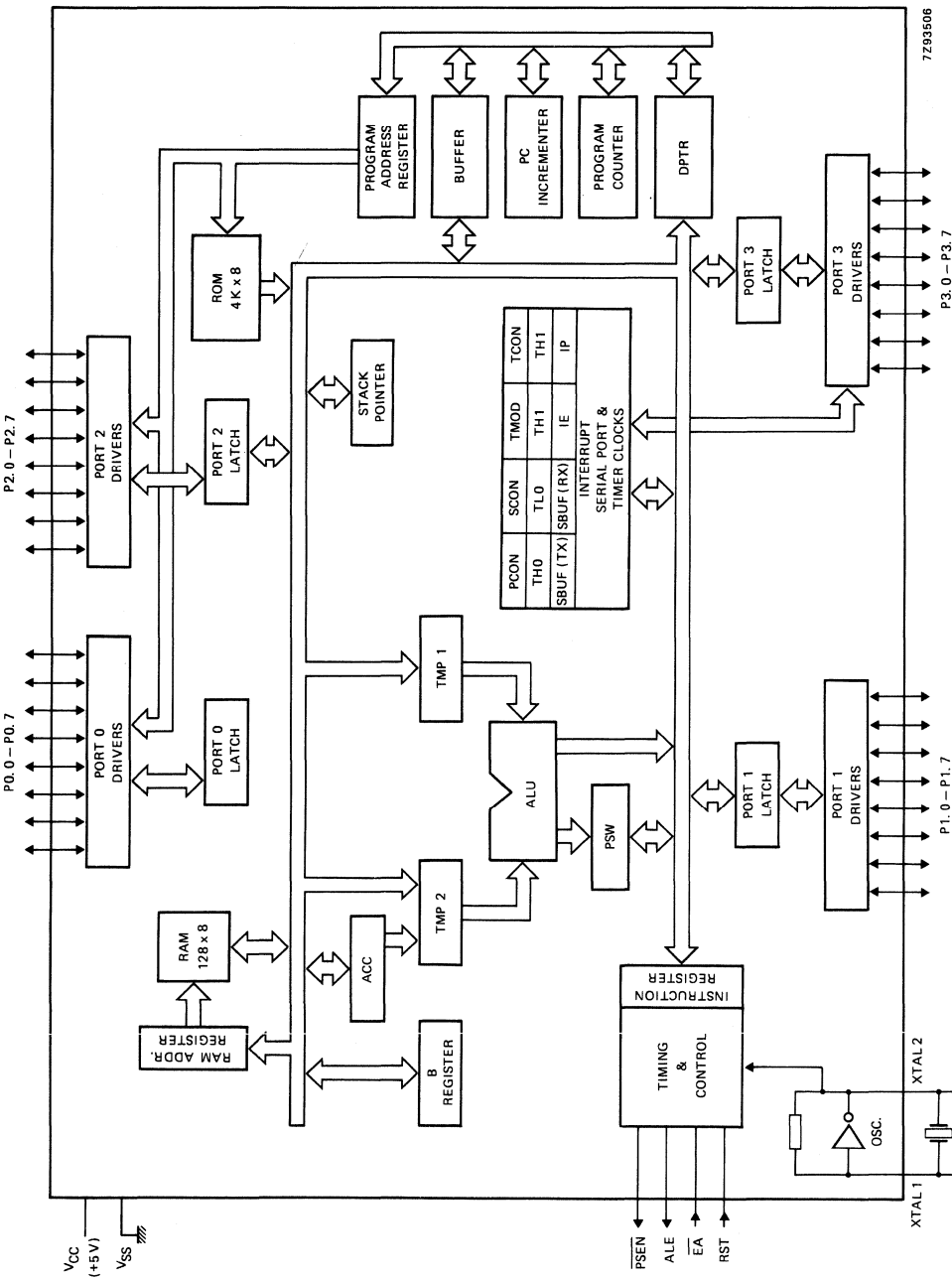


Fig. 1 Block diagram.

DEVELOPMENT DATA

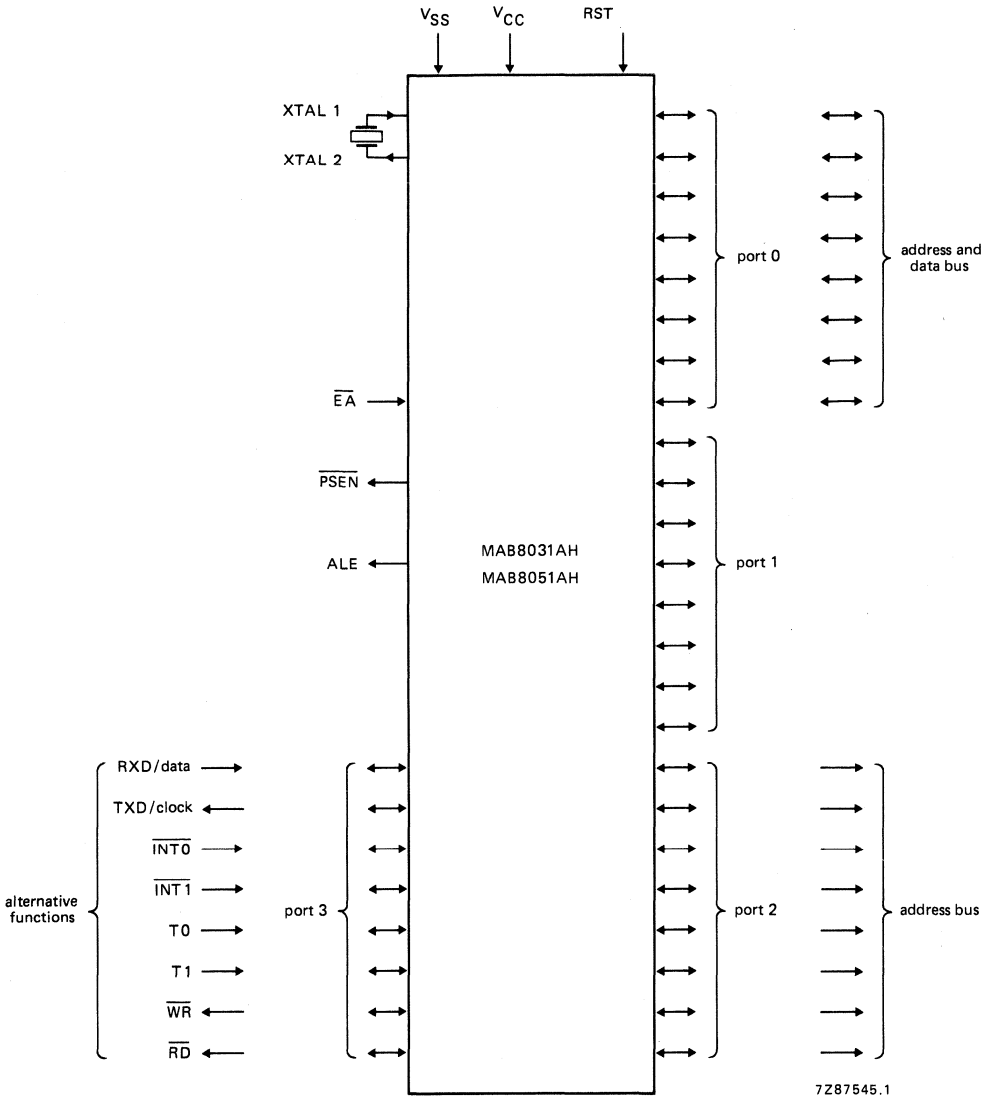


Fig. 2 Functional diagram.

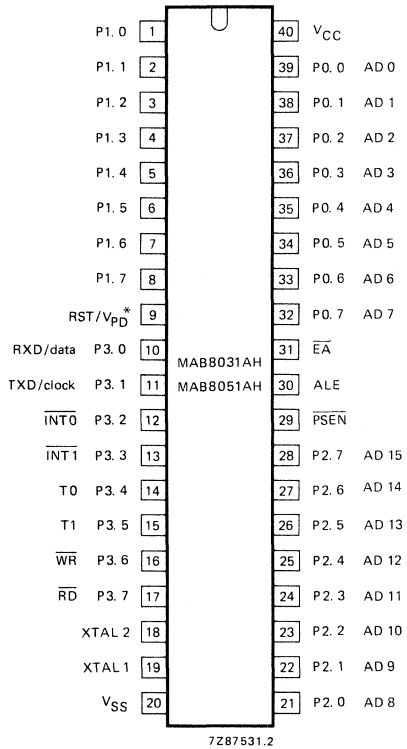


Fig. 3a Pinning diagram for MAB8031/51AHP; MAF8031/51AHP; MAF80A31/51AHP.

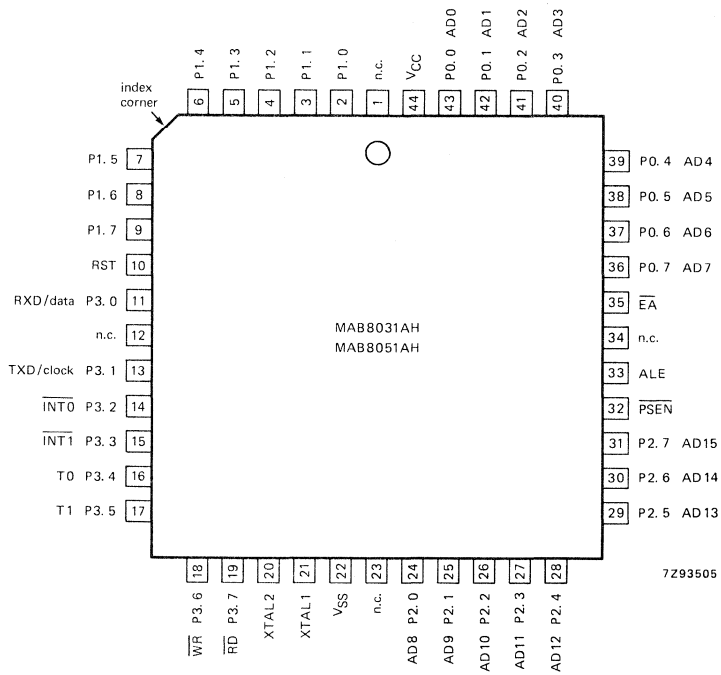


Fig. 3b Pinning diagram for MAB8031/51AHWP; MAF8031/51AHWP; MAF80A31/51AHWP.

* V_{PD} option available on request.

DEVELOPMENT DATA

PINNING (DIL package)

1-8	P1.0-P1.7	**Port 1:** 8-bit quasi-bidirectional I/O port. It receives the low-order address byte during program verification. Port 1 can sink/source one TTL (= 4 LS TTL) input. It can drive MOS inputs without external pull-ups.
9		**RST/V_{PD}**: a high level on this pin for two machine cycles while the oscillator is running resets the device. An internal pulldown permits Power-On reset using only a capacitor connected to V_{CC}. As an available option, this pin also supplies standby power to the RAM: V_{PD} should be held within its specified limit while V_{CC} drops below its specified limit. When V_{PD} is LOW the RAM current is drawn from V_{CC}.
10-17	P3.0-P3.7	**Port 3**: 8-bit quasi-bidirectional I/O port with internal pull-ups. It also serves the following alternative functions:

Port pin	*Alternative function*
P3.0	**RXD/data**: serial port receiver data input (asynchronous) or data input/output (synchronous)
P3.1	**TXD/clock**: serial port transmitter data output (asynchronous) or clock output (synchronous)
P3.2	**$\overline{\text{INT0}}$**: external interrupt 0 or gate control input for timer/event counter 0
P3.3	**$\overline{\text{INT1}}$** : external interrupt 1 or gate control input for timer/event counter 1
P3.4	**T0** : external input for timer/event counter 0
P3.5	**T1** : external input for timer/event counter 1
P3.6	**$\overline{\text{WR}}$**: external data memory write strobe
P3.7	**$\overline{\text{RD}}$** : external data memory read strobe.

Operation of an alternative function is determined by the relevant output latch programmed to logic 1. Port 3 can sink/source one TTL input. It can drive MOS inputs without external pull-ups.

18	XTAL 2	**Crystal input 2**: output of the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external oscillator signal when an external oscillator is used (see figures 6 and 7).
19	XTAL 1	**Crystal input 1**: input to the inverting amplifier that forms the oscillator. Connected to V_{SS} when an external oscillator is used (see figures 6 and 7).
20	V_{SS}	**Ground**: circuit ground potential.
21-28	P2.0-P2.7	**Port 2**: 8-bit quasi-bidirectional I/O port with internal pull-ups. It emits the high-order address byte when accessing external memory. It also receives the high-order address bits and control signals during program verification. Port 2 can sink/source one TTL input. It can drive MOS inputs without external pull-ups.
29	$\overline{\text{PSEN}}$	**Program Store Enable output**: read strobe to the external Program Memory. It is activated twice each machine cycle during fetches from external Program Memory. When executing out of external Program Memory two activations of $\overline{\text{PSEN}}$ are skipped during each access to external Data Memory. $\overline{\text{PSEN}}$ is not activated (remains HIGH) during fetches from internal Program Memory.
30	ALE	**Address Latch Enable output**: latches the low byte of the address during accesses to external memory in normal operation. It is activated every six oscillator periods except during an external data memory access.

PINNING (continued)

31	$\overline{EA}$	When $\overline{EA}$ is held at a TTL high level the CPU executes out of the internal Program Memory (ROM), provided the Program Counter is less than 4096. When $\overline{EA}$ is held at a TTL low level the CPU executes out of external Program Memory. $\overline{EA}$ does not float.
32-39	P0.7-P0.0	**Port 0**: 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus during accesses to external memory (during these accesses it activates internal pull-ups). It also outputs instruction bytes during program verification. (External pull-ups are required during program verification). Port 0 can sink/source two TTL inputs.
40	V_{CC}	**Power Supply**: + 5 V power supply pin during normal operation.

FUNCTIONAL DESCRIPTION

General

The MAB8051AH is a stand-alone high-performance microcontroller designed for use in real-time applications such as instrumentation, industrial control and intelligent computer peripherals.

The device provides hardware features, architectural enhancements and new instructions to function as a controller for applications requiring up to 64 K bytes of program memory and/or up to 64 K bytes of data storage.

The MAB8031AH is a control-oriented CPU without on-chip program memory. It can address 64 K bytes of external program memory in addition to 64 K bytes of external data memory. The MAB8051AH is a MAB8031AH with the lower 4 K bytes of program memory filled with on-chip mask programmable ROM. For systems requiring extra capability, the MAB8051AH can be expanded using standard TTL memories and peripherals.

The two pin-compatible versions of this component reduce development problems to a minimum and provide maximum flexibility. The MAB8051AH is for low-cost, high volume production; and the MAB8031AH for applications requiring the flexibility of external program memory which can be easily modified and updated in the field.

The MAB8051AH contains a non-volatile 4 K x 8 read-only program memory; a volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/event counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and timing circuits.

Central processing unit

The central processing unit (CPU) manipulates operands in four memory spaces. These are the 64 K-byte external data memory, 256-byte internal data memory and 16-bit program counter spaces. The internal data memory address space is sub-divided into the 128-byte internal data RAM and 128-byte special function register (SFR) address spaces, as shown in Fig. 4a. See also Figures 4b to 4f.

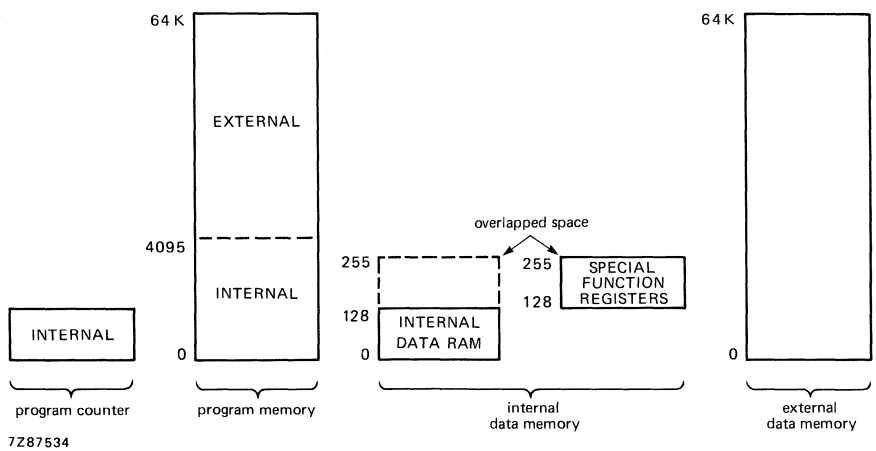


Fig. 4a Memory map.

FUNCTIONAL DESCRIPTION (continued)

The internal data RAM contains four register banks (each with eight registers), 128 addressable bits, and the stack. The stack depth is limited by the available internal data RAM and its location is determined by the 8-bit stack pointer. All registers except the program counter and the four 8-register banks reside in the special function register (SFR) address space. These memory mapped registers include arithmetic registers, pointers, I/O ports, interrupt system registers, timers and serial port. There are 128 addressable bit locations in the SFR address space.

The MAB8051AH contains 128 bytes of internal data RAM and 20 special function registers. It provides a non-paged program memory address space to accommodate relocatable code. Conditional branches are performed relative to the program counter. The register-indirect jump permits branching relative to a 16-bit base register with an offset provided by an 8-bit index register. 16-bit jumps and calls permit branching to any location in the contiguous 64 K program memory address space.

The MAB8051AH has five methods for addressing source operands:

- Register
- Direct
- Register-Indirect
- Immediate
- Base-Register-plus Index-Register-Indirect

The first three methods can be used for addressing destination operands. Most instructions have a "destination source" field that specifies the data type, addressing methods and operands involved. For operations other than moves, the destination operand is also a source operand.

Access addressing is as follows:

- Registers in the four 8-register banks through Register, Direct, or Register-Indirect.
- 128 bytes of internal data RAM through Direct or Register-Indirect.
- Special function registers through Direct.
- External data memory through Register-Indirect.
- Program memory look-up tables through Base-Register-plus Index-Register-Indirect.

The MAB8051AH is classified as an 8-bit device since the internal ROM, RAM, Special Function Registers (SFR), Arithmetic Logic Unit (ALU), and external data bus are each 8-bits wide. It performs operations on bit, nibble, byte and double-byte data types.

Facilities are available for byte transfer, logic, and integer arithmetic operations. Data transfer, logic, and conditional branch operations can be performed directly on Boolean variables to provide excellent bit handling.

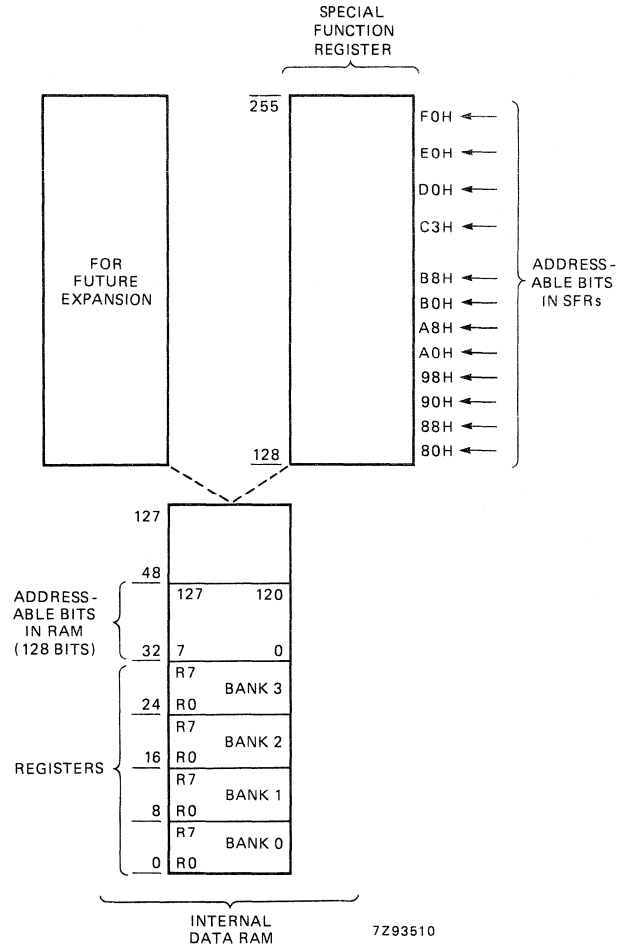


Fig. 4b Internal data memory address space.

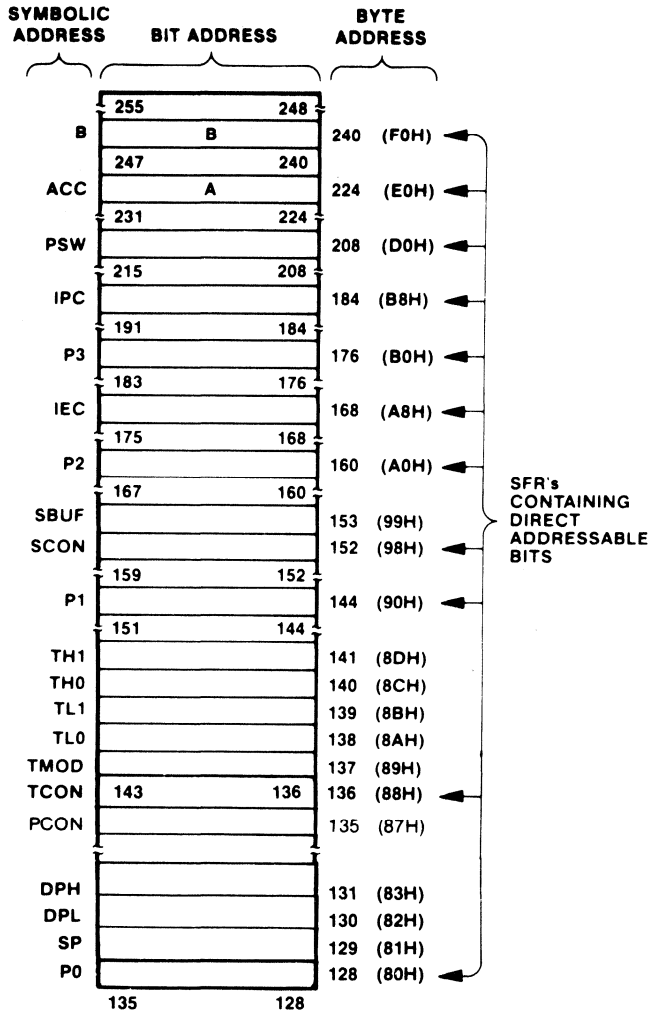


Fig. 4c Mapping of special function registers.

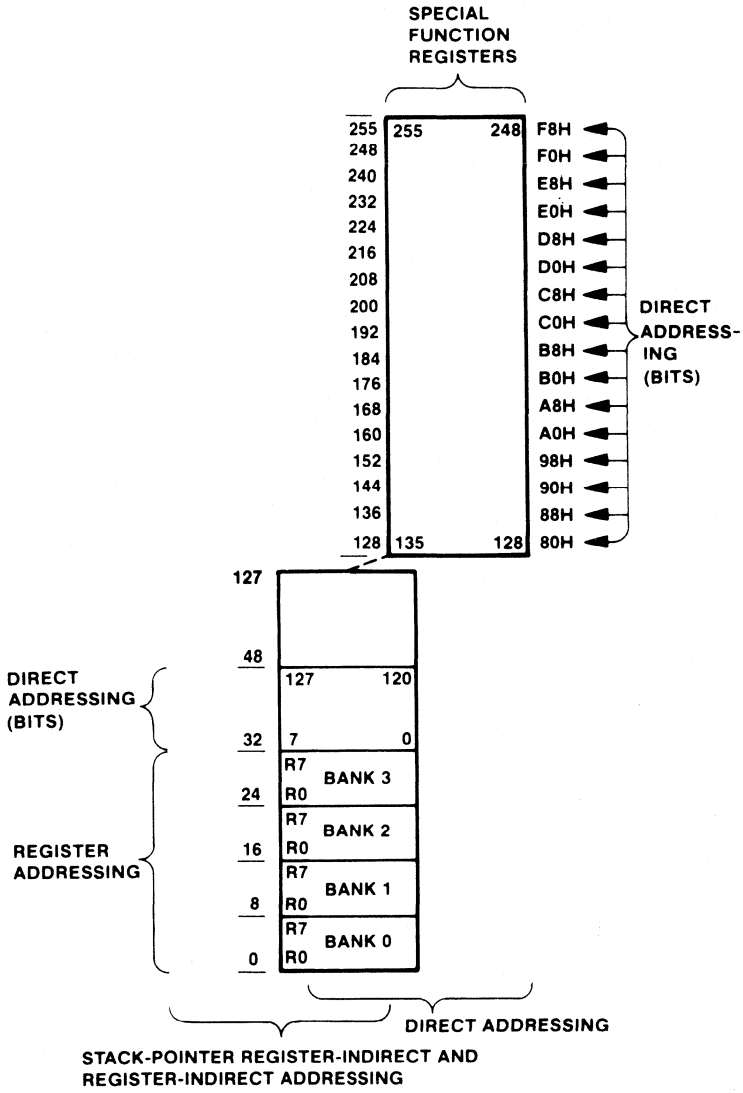


Fig. 4d Special function register bit addresses.

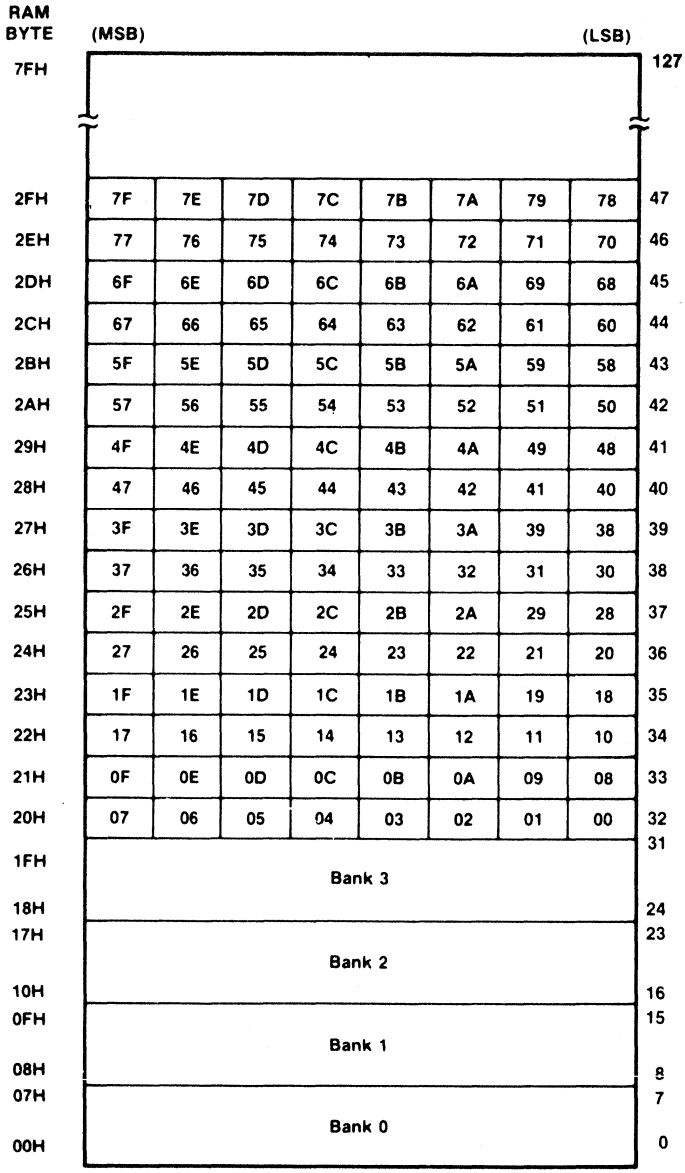


Fig. 4e RAM bit addresses.

DEVELOPMENT DATA

Direct Byte Address	Bit Addresses (MSB)							(LSB)	Hardware Register Symbol
240	F7	F6	F5	F4	F3	F2	F1	F0	B
224	E7	E6	E5	E4	E3	E2	E1	E0	ACC
	CY	AC	FO	RS1	RS0	OV		P	
208	D7	D6	D5	D4	D3	D2	D1	D0	PSW
				PS	PT1	PX1	PT0	PX0	
184	—	—	—	BC	BB	BA	B9	B8	IP
176	B7	B6	B5	B4	B3	B2	B1	B0	P3
	EA			ES	ET1	EX1	ET0	EX0	
168	AF	—	—	AC	AB	AA	A9	A8	IE
160	A7	A6	A5	A4	A3	A2	A1	A0	P2
	SMO	SM1	SM2	REN	TB8	RB8	TI	RI	
152	9F	9E	9D	9C	9B	9A	99	98	SCON
144	97	96	95	94	93	92	91	90	P1
	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0	
136	8F	8E	8D	8C	8B	8A	89	88	TCON
128	87	86	85	84	83	82	81	80	P0

Fig. 4f Addressing operands in internal data memory.

FUNCTIONAL DESCRIPTION (continued)

I/O facilities

The MAB8051AH has 32 I/O lines treated as 32 individual addressable bits and as four parallel 8-bit addressable ports. Ports 0, 1, 2 and 3 perform the following alternate functions:

- Port 0; provides the multiplexed low-order address and data bus used for expanding the MAB8051AH with standard memories and peripherals
- Port 2; provides the high-order address bus when expanding the MAB8051 with external program memory or more than 256 bytes of external data memory
- Port 3; pins can be configured individually to provide:-

  external interrupt requests inputs
  counter inputs
  serial port receiver input and transmitter output
  control signals to READ and WRITE to external data memory

The generation or use of a Port 3 pin as an alternate function is carried out automatically by the MAB8051AH provided the pin is configured as an output.

Timer/event counters

The MAB8051AH contains two 16-bit registers, Timer 0 and Timer 1, that can be used as timers or event counters to carry out the following functions:

- Measure time intervals and pulse durations
- Count events
- Generate interrupt requests

Each timer/event counter can be programmed independently to operate in three modes:

- Mode 0; 8-bit timer or 8-bit counter each with divide by 32 prescaler
- Mode 1; 16-bit time-interval or event counter
- Mode 2; 8-bit time-interval or event counter with automatic reload upon overflow

Counter 0 can be programmed to operate in an additional mode as follows:

- Mode 3; one 8-bit time-interval or event counter and one 8-bit time-interval counter

When counter 0 is in Mode 3, counter 1 can be programmed to operate in Modes 0, 1 or 2 but cannot set an interrupt request flag or generate an interrupt. However the overflow from counter 1 can be used to pulse the serial Port transmission-rate generator.

The frequency handling range of these counters with a 3,5 to 12 MHz crystal is as follows:

- 0,3 to 1 MHz when programmed for an input that is a division by 12 of the oscillator frequency
- 0 Hz to an upper limit of 150 kHz to 0,5 MHz when programmed for external inputs

Both internal and external inputs can be gated to the counter by a second external source for directly measuring pulse durations.

The counters are started and stopped under software control. Each one sets its interrupt request flag when it overflows from all 1's to all 0's (or automatic reload value).

On-chip peripheral functions

In addition to the CPU and memories, an interrupt system, extensive I/O facilities, and several peripheral functions are integrated on-chip to relieve the CPU of repetitious, complicated or time-critical tasks and to permit stringent real-time control of external system interfaces. The I/O facilities include the I/O pins, parallel ports, bidirectional address/data bus and the serial port for I/O expansion. The CPU peripheral functions integrated on-chip are the two 16-bit timer/event counters and the serial port.

Interrupt system (see Fig. 5)

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution a multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response latency is from 3 μs to 7 μs when using a 12 MHz crystal.

The MAB8051AH acknowledge interrupt requests from five sources as follows:

- $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$; externally via pins 12 and 13 respectively
- Timer 0 and Timer 1; from the two internal counters
- Serial Port; from the internal serial I/O port

Each interrupt vectors to a separate location in program memory for its service program. Each source can be individually enabled or disabled and can be programmed to a high or low priority level. Also all enabled sources can be globally disabled or enabled. Both external interrupts can be programmed to be level-activated or transition-activated and is active LOW to allow "wire-ORing" of several interrupt sources to the input pin.

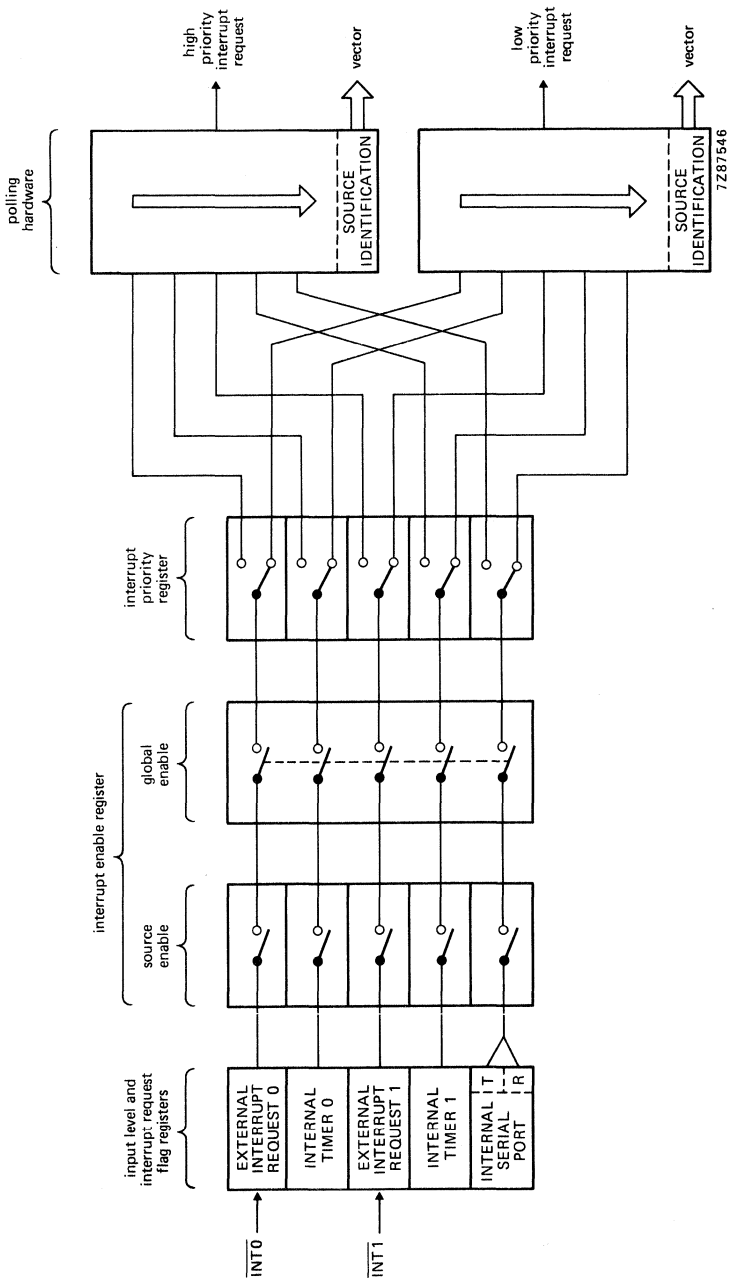


Fig. 5 Interrupt system.

OSCILLATOR CIRCUITRY

The oscillator circuitry of the MAB8051AH is a single-stage inverting amplifier in a Pierce oscillator configuration. The circuitry has a combination of depletion and enhancement mode MOS FETs to produce the inverting characteristics, and not passive components. Either a crystal or ceramic resonator can be used as the feedback element to complete the oscillator circuitry. XTAL1, pin 19, is the high gain amplifier input, and XTAL2, pin 18, is the output (see Fig. 6).

To drive the MAB8051AH externally, XTAL1 should be connected to ground and XTAL2 driven from an external source (see Fig. 7).

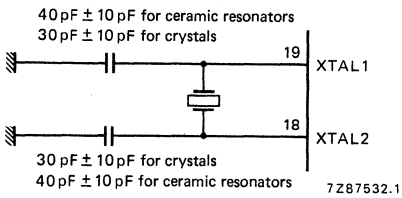


Fig. 6 MAB8051AH oscillator circuit.

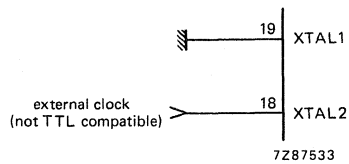


Fig. 7 Driving the MAB8051AH from an external source.

DEVELOPMENT DATA

RESET CIRCUITRY

The reset circuitry for the MAB8051AH is connected to the reset pin, RST, as shown in Fig. 8. A Schmitt trigger is used th the input for noise rejection. The output of the Schmitt trigger is sampled by the reset circuitry every machine cycle.

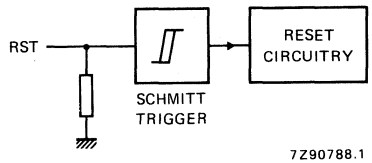


Fig. 8 Reset configuration at RST.

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods), while the oscillator is running. The CPU responds by executing an internal reset. It also configures the ALE and $\overline{PSEN}$ pins as inputs. (They are quasi-bidirectional.) The internal reset is executed during the second cycle in which RST is high and is repeated every cycle until RST goes LOW. It leaves the internal registers as follows:

REGISTER	CONTENT
PC	0000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0 -- P3	0FFH
IP	XXX00000B
IE	XXX00000B
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	Indeterminate
PCON	0XXXXXXXB

Where

H = Hexadecimal
B = Binary

The internal RAM is not affected by reset. When VCC is turned on, the RAM content is determinate.

RESET CIRCUITRY (continued)

Power-on reset

An automatic reset when V_{CC} is turned on can be obtained by connecting the RST pin to V_{CC} through a 10 μF capacitor, as long as the V_{CC} rise-time does not exceed 10 milliseconds. This power-on reset circuit is shown in Fig. 9. When the power is switched on, the current drawn by RST is the difference between V_{CC} and the capacitor voltage, and decreases from V_{CC} as the capacitor charges. The larger the capacitor, the more slowly V_{RST} decreases. V_{RST} must remain above the lower threshold of the Schmitt trigger long enough to effect a complete reset. The time required is the oscillator start-up time, plus 2 machine cycles.

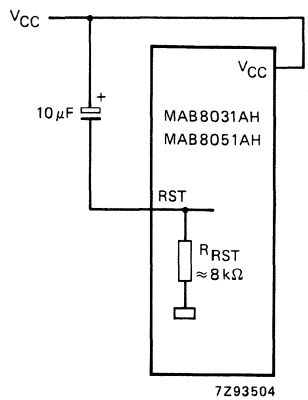


Fig. 9 Power-on reset.

INSTRUCTION SET

The MAB8051AH uses a powerful instruction set to allow expansion of on-chip CPU peripherals and to optimize byte efficiency and execution speed. Reassigned opcodes add new high-power operations and permit new addressing modes to make old operations more orthogonal. The instruction set consists of 49 single-byte, 45 two-byte and 17 three-byte instructions. When using a 12 MHz oscillator, 64 instructions execute in 1 μs and 45 instructions execute in 2 μs. Multiply and divide instructions execute in 4 μs.

Table 1 Instruction set description

mnemonic		description	bytes/ cycles	opcode (hex.)
Arithmetic operation				
ADD	A,Rr	Add register to A	1 1	2*
ADD	A,direct	Add direct byte to A	2 1	25
ADD	A,@Ri	Add indirect RAM to A	1 1	26, 27
ADD	A,#data	Add immediate data to A	2 1	24
ADDC	A,Rr	Add register to A with carry flag	1 1	3*
ADDC	A,direct	Add direct byte to A with carry flag	2 1	35
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1 1	36, 37
ADDC	A,#data	Add immediate data to A with carry flag	2 1	34
SUBB	A,Rr	Subtract register from A with borrow	1 1	9*
SUBB	A,direct	Subtract direct byte from A with borrow	2 1	95
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1 1	96, 97
SUBB	A,#data	Subtract immediate data from A with borrow	2 1	94
INC	A	Increment A	1 1	04
INC	Rr	Increment register	1 1	0*
INC	direct	Increment direct byte	2 1	05
INC	@Ri	Increment indirect RAM	1 1	06, 07
DEC	A	Decrement A	1 1	14
DEC	Rr	Decrement register	1 1	1*
DEC	direct	Decrement direct byte	2 1	15
DEC	@Ri	Decrement indirect RAM	1 1	16, 17
INC	DPTR	Increment data pointer	1 2	A3
MUL	AB	Multiply A & B	1 4	A4
DIV	AB	Divide A by B	1 4	84
DA	A	Decimal adjust A	1 1	D4

mnemonic		description	bytes/ cycles	opcode (hex.)
Logic operations				
ANL	A,Rr	AND register to A	1 1	5*
ANL	A,direct	AND direct byte to A	2 1	55
ANL	A,@Ri	AND indirect RAM to A	1 1	56, 57
ANL	A,#data	AND immediate data to A	2 1	54
ANL	direct,A	AND A to direct byte	2 1	52
ANL	direct,#data	AND immediate data to direct byte	3 2	53
ORL	A,Rr	OR register to A	1 1	4*
ORL	A,direct	OR direct byte to A	2 1	45
ORL	A,@Ri	OR indirect RAM to A	1 1	46, 47
ORL	A,#data	OR immediate data to A	2 1	44
ORL	direct,A	OR A to direct byte	2 1	42
ORL	direct,#data	OR immediate data to direct byte	3 2	43
XRL	A,Rr	Exclusive-OR register to A	1 1	6*
XRL	A,direct	Exclusive-OR direct byte to A	2 1	65
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1 1	66, 67
XRL	A,#data	Exclusive-OR immediate data to A	2 1	64
XRL	direct, A	Exclusive-OR to direct byte	2 1	62
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3 2	63
CLR	A	Clear A	1 1	E4
CPL	A	Complement A	1 1	F4
RL	A	Rotate A left	1 1	23
RLC	A	Rotate A left through the carry flag	1 1	33
RR	A	Rotate A right	1 1	03
RRC	A	Rotate A right through the carry flag	1 1	13
SWAP	A	Swap nibbles within A	1 1	C4

INSTRUCTION SET (continued)

mnemonic		description	bytes/ cycles	opcode (hex.)
Data transfer				
MOV	A,Rr	Move register to A	1 1	E*
MOV	A,direct	Move direct byte to A	2 1	E5
MOV	A,@Ri	Move indirect RAM to A	1 1	E6, E7
MOV	A,#data	Move immediate data to A	2 1	74
MOV	Rr,A	Move A to register	1 1	F*
MOV	Rr,direct	Move direct byte to register	2 2	A*
MOV	Rr,#data	Move immediate data to register	2 1	7*
MOV	direct,A	Move A to direct byte	2 1	F5
MOV	direct,Rr	Move register to direct byte	2 2	8*
MOV	direct,direct	Move direct byte to direct	3 2	85
MOV	direct,@Ri	Move indirect RAM to direct byte	2 2	86, 87
MOV	direct,#data	Move immediate data to direct byte	3 2	75
MOV	@Ri,A	Move A to indirect RAM	1 1	F6, F7
MOV	@Ri,direct	Move direct byte to indirect RAM	2 2	A6, A7
MOV	@Ri,#data	Move immediate data to indirect RAM	2 1	76, 77
MOV	DPTR,#data16	Load data pointer with a 16-bit constant	3 2	90
MOVC	A,@A+DPTR	Move code byte relative to DPTR to A	1 2	93
MOVC	A,@A+PC	Move code byte relative to PC to A	1 2	83
MOVX	A,@Ri	Move external RAM (8-bit address) to A	1 2	E2, E3
MOVX	A,@DPTR	Move external RAM (16-bit address) to A	1 2	E0
MOVX	@Ri,A	Move A to external RAM (8-bit address)	1 2	F2, F3
MOVX	@DPTR,A	Move A to external RAM (16-bit address)	1 2	F0
PUSH	direct	Push direct byte onto stack	2 2	C0
POP	direct	Pop direct byte from stack	2 2	D0
XCH	A,Rr	Exchange register with A	1 1	C*
XCH	A,direct	Exchange direct byte with A	2 1	C5
XCH	A,@Ri	Exchange indirect RAM with A	1 1	C6, C7
XCHD	A,@Ri	Exchange LOW-order digit indirect RAM with A	1 1	D6, D7

mnemonic		description	bytes/ cycles	opcode (hex.)
Boolean variable manipulation				
CLR	C	Clear carry flag	1 1	C3
CLR	bit	Clear direct bit	2 1	C2
SETB	C	Set carry flag	1 1	D3
SETB	bit	Set direct bit	2 1	D2
CPL	C	Complement carry flag	1 1	B3
CPL	bit	Complement direct bit	2 1	B2
ANL	C,bit	AND direct bit to carry flag	2 2	82
ANL	C,/bit	AND complement of direct bit to carry flag	2 2	B0
ORL	C,bit	OR direct bit to carry flag	2 2	72
ORL	C,/bit	OR complement of direct bit to carry flag	2 2	A0
MOV	C,bit	Move direct bit to carry flag	2 1	A2
MOV	bit,C	Move carry flag to direct bit	2 2	92
Program and machine control				
ACALL	addr11	Absolute subroutine call	2 2	●1addr
LCALL	addr16	Long subroutine call	3 2	12
RET		Return from subroutine	1 2	22
RET1		Return from interrupt	1 2	32
AJMP	addr11	Absolute jump	2 2	▲1addr
LJMP	addr16	Long jump	3 2	02
SJMP	rel	Short jump (relative address)	2 2	80
JMP	@A+DPTR	Jump indirect relative to the DPTR	1 2	73
JZ	rel	Jump if A is zero	2 2	60
JNZ	rel	Jump if A is not zero	2 2	70
JC	rel	Jump if carry flag is set	2 2	40
JNC	rel	Jump if no carry flag	2 2	50
JB	bit,rel	Jump if direct bit is set	3 2	20
JNB	bit,rel	Jump if direct bit is not set	3 2	30
JBC	bit,rel	Jump if direct bit is set and clear bit	3 2	10
CJNE	A,direct,rel	Compare direct to A and jump if not equal	3 2	B5
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3 2	B4
CJNE	Rr,#data,rel	Compare immed. to reg. and jump if not equal	3 2	B*
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3 2	B6, B7
DJNZ	Rr,rel	Decrement register and jump if not zero	2 2	D*
DJNZ	direct,rel	Decrement direct and jump if not zero	3 2	D5
NOP		No operation	1 1	00

Notes to Table 1

Data addressing modes

Rr	Working register R0-R7.
direct	128 internal RAM locations and any special function register (SFR).
@Ri	Indirect internal RAM location addressed by register R0 or R1.
#data	8-bit constant included in instruction.
#data16	16-bit constant included as bytes 2 and 3 of instruction.
bit	Direct addressed bit in internal RAM or SFR.
addr16	16-bit destination address. Used by LCALL and LJMP. The branch will be anywhere within the 64 K-byte program memory address space.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2 K-byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to + 127 bytes relative to first byte of the following instruction.

Hexadecimal opcode cross-reference to Table 2

*: 8, 9, A, B, C, D, E, F.

●: 11, 31, 51, 71, 91, B1, D1, F1.

▲: 01, 21, 41, 61, 81, A1, C1, E1.

Table 2 Instruction map

first hexadecimal character of opcode ↓	second hexadecimal character of opcode: 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	AJMP page 0	LJMP addr16	RR A	INC A	INC dir	INC @Ri 0	1	INC Rr 0	1	2	3	4	5	6	7
1	JBC bit, addr8	ACALL page 0	LCALL addr16	RRC A	DEC A	DEC dir	DEC @Ri 0	1	DEC Rr 0	1	2	3	4	5	6	7
2	JB bit, addr8	AJMP page 1	RET	RL A	ADD A,#data	ADD A,dir	ADD A,@Ri 0	1	ADD A,Rr 0	1	2	3	4	5	6	7
3	JNB bit, addr8	ACALL page 1	RET1	RLC A	ADDC A,#data	ADDC A,dir	ADDC A,@Ri 0	1	ADDC A,Rr 0	1	2	3	4	5	6	7
4	JC addr8	AJMP page 2	ORL dir,A	ORL dir,#data	ORL A,#data	ORL A,dir	ORL A,@Ri 0	1	ORL A,Rr 0	1	2	3	4	5	6	7
5	JNC addr8	ACALL page 2	ANL dir,A	ANL dir,#data	ANL A,#data	ANL A,dir	ANL A,@Ri 0	1	ANL A,Rr 0	1	2	3	4	5	6	7
6	JZ addr8	AJMP page 3	XRL dir,A	XRL dir,#data	XRL A,#data	XRL A,dir	XRL A,@Ri 0	1	XRL A,Rr 0	1	2	3	4	5	6	7
7	JNZ addr8	ACALL page 3	ORL C,bit	JMP @A+DPTR	MOV A,#data	MOV dir,#data	MOV @Ri,#data 0	1	MOV Rr,#data 0	1	2	3	4	5	6	7
8	SJMP addr8	AJMP page 4	ANL C,bit	MOVC A,@A+PC	DIV AB	MOV dir,dir	MOV dir,@Ri 0	1	MOV dir,Rr 0	1	2	3	4	5	6	7
9	MOV DPTR, #data 16	ACALL page 4	MOV bit,C	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,dir	SUBB A,@Ri 0	1	SUBB A,Rr 0	1	2	3	4	5	6	7
A	ORL C,/bit	AJMP page 5	MOV C,bit	INC DPTR	MUL AB		MOV @Ri,dir 0	1	MOV Rr,dir 0	1	2	3	4	5	6	7
B	ANL C,/bit	ACALL page 5	CPL bit	CPL C	CJNE A, #data, addr8	CJNE A,dir, addr8	CJNE @Ri,#data, addr8 0	1	CJNE Rr,#data, addr8 0	1	2	3	4	5	6	7
C	PUSH dir	AJMP page 6	CLR bit	CLR C	SWAP A	XCH A,dir	XCH A,@Ri 0	1	XCH A,Rr 0	1	2	3	4	5	6	7
D	POP dir	ACALL page 6	SETB bit	SETB C	DA A	DJNZ dir, addr8	XCHD A,@Ri 0	1	DJNZ Rr, addr8 0	1	2	3	4	5	6	7
E	MOVX A,@DPTR	AJMP page 7	MOVX A,@Ri 0	1	CLR A	MOV A,dir	MOV A,@Ri 0	1	MOV A,Rr 0	1	2	3	4	5	6	7
F	MOVX @DPTR,A	ACALL page 7	MOVX @Ri,A 0	1	CPL A	MOV dir,A	MOV @Ri,A 0	1	MOV Rr,A 0	1	2	3	4	5	6	7

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Input voltage on any pin with respect to ground (V_{SS})	V_I	−0,5 to + 7	V
Total power dissipation	P_{tot}	max. 1	W
Input, output current	$\pm I_I, I_O$	max. 10	mA
Storage temperature range	T_{stg}	−65 to + 150	°C
Operating ambient temperature range			
MAB8031/51AH	T_{amb}	0 to + 70	°C
MAF8031/51AH	T_{amb}	−40 to + 85	°C
MAF80A31/51AH	T_{amb}	−40 to + 100	°C

D.C. CHARACTERISTICS (MAB8031/51AH)

V_{CC} = 5 V (± 10%); V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	max.	unit	conditions
Supply current	I_{CC}	–	160	mA	
Supply current (optional)	I_{PD}	–	20	mA	
Inputs					
Input voltage LOW	V_{IL}	−0,5	0,8	V	
Input voltage HIGH all inputs except RST and XTAL 2	V_{IH}	2	V_{CC} + 0,5	V	
Input voltage HIGH to RST and XTAL 2	V_{IH1}	2,5		V	XTAL 1 to V_{SS}
Outputs					
Output voltage LOW (Ports 1, 2, 3) (note 1)	V_{OL}		0,45	V	I_{OL} = 1,6 mA
Output voltage LOW (Port 0, ALE, $\overline{PSEN}$) (note 1)	V_{OL1}		0,45	V	I_{OL1} = 3,2 mA
Output voltage HIGH (Port 1, 2, 3)	V_{OH}	2,4		V	I_{OH} = −80 μA
Output voltage HIGH (Port 0, ALE, $\overline{PSEN}$)	V_{OH1}	2,4		V	I_{OH1} = −400 μA
Input leakage current (Port 0, $\overline{EA}$)	$\pm I_{LI}$		10	μA	$V_{SS} < V_I < V_{CC}$
Input current HIGH (RST)	I_{IH1}		500	μA	$V_I = V_{CC}$−1,5 V
current logic 0 (Ports 1, 2, 3)	I_{IL}		−800	μA	V_{IL} = 0,45 V
Input current logic 0 (XTAL 2)	I_{IL2}		−2,5	mA	V_{IL} = 0,45 V; XTAL 1 to V_{SS}
Capacitance of I/O buffer	$C_{I/O}$		10	pF	f_c = 1 MHz; T_{amb} = 25 °C

DEVELOPMENT DATA

D.C. CHARACTERISTICS (MAF8031/51AH; MAF80A31/51AH)

V_{CC} = 5 V (± 10%); V_{SS} = 0 V; T_{amb} = −40 to + 85 °C (MAF8031/51AH), −40 to + 100 °C (MAF80A31/51AH); all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	max.	unit	conditions
Supply current	I_{CC}	–	175	mA	
Inputs					
Input voltage LOW	V_{IL}	−0,5	0,8	V	
Input voltage HIGH all inputs except RST and XTAL 2	V_{IH}	2,2	V_{CC} + 0,5	V	
Input voltage HIGH to RST and XTAL 2	V_{IH1}	2,7		V	XTAL 1 to V_{SS}
Outputs					
Output voltage LOW (Ports 1, 2, 3) (note 1)	V_{OL}		0,45	V	I_{OL} = 1,2 mA
Output voltage LOW (Port 0, ALE, $\overline{PSEN}$) (note 1)	V_{OL1}		0,45	V	I_{OL1} = 2,4 mA
Output voltage HIGH (Port 1, 2, 3)	V_{OH}	2,4		V	I_{OH} = −50 μA
Output voltage HIGH (Port 0, ALE $\overline{PSEN}$)	V_{OH1}	2,4		V	I_{OH1} = −360 μA
Input leakage current (Port 0, $\overline{EA}$)	± I_{LI}		10	μA	$V_{SS} < V_I < V_{CC}$
Input current HIGH (RST)	I_{IH1}		500	μA	$V_I = V_{CC}$−1,5 V
current logic 0 (Ports 1, 2, 3)	I_{IL}		−800	μA	V_{IL} = 0,45 V
Input current logic 0 (XTAL 2)	I_{IL2}		−3,0	mA	V_{IL} = 0,45 V; XTAL 1 to V_{SS}
Capacitance of I/O buffer	$C_{I/O}$		10	pF	f_c = 1 MHz; T_{amb} = 25 °C

Note 1

V_{OL} is degraded when the MAB8051AH rapidly discharges external capacitance.
This a.c. noise is most pronounced during emission of address data. When using external memory, locate the latch or buffer as close to the MAB8051AH as possible.

datum	emitting ports	time slot interval	degraded I/O lines	V_{OL} (max.)
address	P2, P0	TS3, TS9	P1, P3	0,8 V
write data	P0	TS6	P1, P3, ALE	0,8 V

A.C. CHARACTERISTICS (note 1)

V_{CC} = 5 V ± 10%; V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; C_L = 100 pF (Port 0, ALE and $\overline{PSEN}$); C_L = 80 pF all other outputs unless otherwise specified (see waveforms Figs 7, 8 and 9).

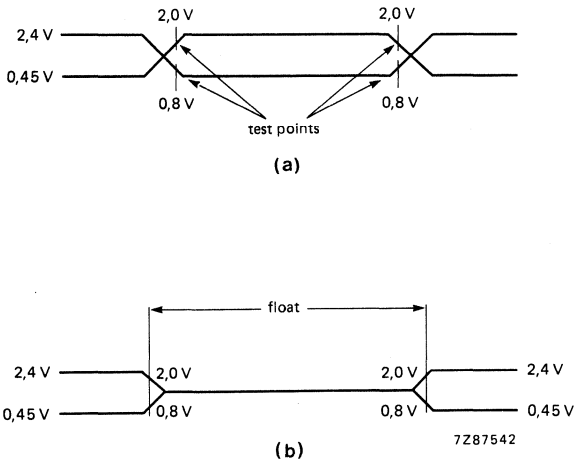
parameter	symbol	8 MHz		10 MHz		12 MHz		variable clock (note 2)		unit
		min.	max.	min.	max.	min.	max.	min.	max.	
Program memory										
ALE pulse duration	t_{LL}	127	–	160	–	127	–	$2t_{CK}-40$	–	ns
Address set-up time to ALE	t_{AL}	85	–	60	–	43	–	$t_{CK}-40$	–	ns
Address hold time after ALE	t_{LA}	90	–	65	–	48	–	$t_{CK}-35$	–	ns
Time from ALE to valid instruction input	t_{LIV}	–	400	–	300	–	233	–	$4t_{CK}-100$	ns
Time from ALE to control pulse $\overline{PSEN}$	t_{LC}	100	–	75	–	58	–	$t_{CK}-25$	–	ns
Control pulse duration $\overline{PSEN}$	t_{CC}	340	–	265	–	215	–	$3t_{CK}-35$	–	ns
Time from $\overline{PSEN}$ to valid instruction input	t_{CIV}	–	250	–	175	–	125	–	$3t_{CK}-125$	ns
Input instruction hold time after $\overline{PSEN}$	t_{CI}	0	–	0	–	0	–	0	–	ns
Input instruction float delay after $\overline{PSEN}$ (note 3)	t_{CIF}	–	105	–	80	–	63	–	$t_{CK}-20$	ns
Address valid after $\overline{PSEN}$ (note 3)	t_{AC}	117	–	92	–	75	–	$t_{CK}-8$	–	ns
Address to valid instruction input	t_{AIV}	–	510	–	385	–	302	–	$5t_{CK}-115$	ns
Address float time to $\overline{PSEN}$	t_{AFC}	−12	–	−12	–	−12	–	−12	–	ns

DEVELOPMENT DATA

parameter	symbol	8 MHz		10 MHz		12 MHz		variable clock (note 2)		unit
		min.	max.	min.	max.	min.	max.	min.	max.	
External data memory										
$\overline{RD}$ pulse duration	t_{RR}	650	–	500	–	400	–	$6t_{CK}-100$	–	ns
$\overline{WR}$ pulse duration	t_{WW}	650	–	500	–	400	–	$6t_{CK}-100$	–	ns
Address hold time after ALE	t_{LA}	90	–	65	–	48	–	$t_{CK}-35$	–	ns
RD to valid data input	t_{RD}	–	460	–	335	–	250	–	$5t_{CK}-165$	ns
Data hold time after $\overline{RD}$	t_{DR}	0	–	0	–	0	–	0	–	ns
Data float delay after $\overline{RD}$	t_{DFR}	–	180	–	130	–	97	–	$2t_{CK}-70$	ns
Time from ALE to valid data input	t_{LD}	–	850	–	650	–	517	–	$8t_{CK}-150$	ns
Address to valid data input	t_{AD}	–	960	–	735	–	585	–	$9t_{CK}-165$	ns
Time from ALE to $\overline{RD}$ or $\overline{WR}$	t_{LW}	325	425	250	350	200	300	$3t_{CK}-50$	$3t_{CK}+50$	ns
Time from address to $\overline{RD}$ or $\overline{WR}$	t_{AW}	370	–	270	–	203	–	$4t_{CK}-130$	–	ns
Time from $\overline{RD}$ or $\overline{WR}$ HIGH to ALE HIGH	t_{WHLH}	85	165	60	140	43	123	$t_{CK}-40$	$t_{CK}+40$	ns
Data valid to $\overline{WR}$ transition	t_{DWX}	65	–	40	–	23	–	$t_{CK}-60$	–	ns
Data set-up time before $\overline{WR}$	t_{DW}	725	–	550	–	433	–	$7t_{CK}-150$	–	ns
Data hold time after $\overline{WR}$	t_{WD}	75	–	50	–	33	–	$t_{CK}-50$	–	ns
Address float delay after $\overline{RD}$	t_{AFR}	–	−12	–	−12	–	−12	–	−12	ns

Notes to the a.c. characteristics

1. T_{amb} for MAF8031AH/MAF8051AH is −40 to 85 °C;
   for MAF80A31AH/MAF80A51AH is −40 to 100 °C.
   The clock frequencies of 8 MHz and 10 MHz applies to all devices; 12 MHz to MAB8051AH and MAF80A31AH; 15 MHz to MAB8031AH.
2. $1/t_{CK}$ = 3,5 to 15 MHz (see Fig. 11 and Table 3).
3. Interfacing the MAB8051AH to devices with float times up to 75 ns is permitted. This limited bus contention will not cause damage to port 0 drivers.



A.C. testing inputs are driven at 2,4 V for a logic 1 and 0,45 V for a logic 0. Timing measurements are taken at 2,0 V for a logic 1 and 0,8 V for logic 0. The float state is defined as the point at which a Port 0 pin sinks 3,2 mA or sources 400 μA at the voltage test levels.

Fig. 10 A.C. testing input, output waveform (a) and float waveform (b).

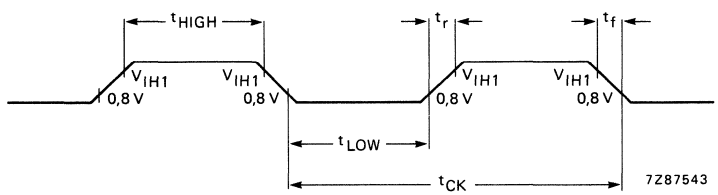


Fig. 11 External clock drive XTAL 2 (see Table 3).

Table 3 External clock drive XTAL 2 (see Fig. 11)

parameter	symbol	variable clock (f = 3,5 to 15 MHz)		unit
		min.	max.	
oscillator clock period	t_{CK}	66,6	286	ns
HIGH time	t_{HIGH}	20	$t_{CK}-t_{LOW}$	ns
LOW time	t_{LOW}	20	$t_{CK}-t_{HIGH}$	ns
rise time	t_r	–	20	ns
fall time	t_f	–	20	ns

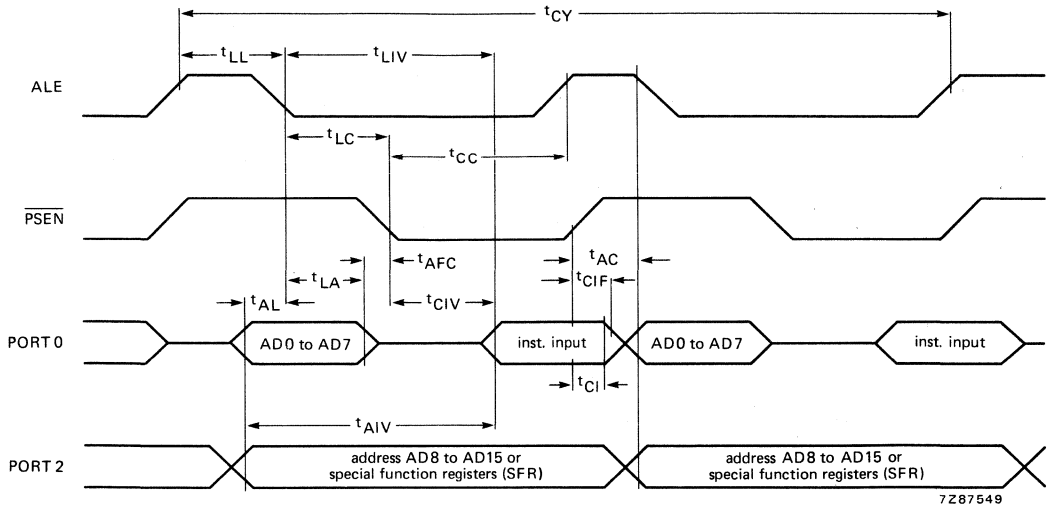


Fig. 12 Read from program memory.

DEVELOPMENT DATA

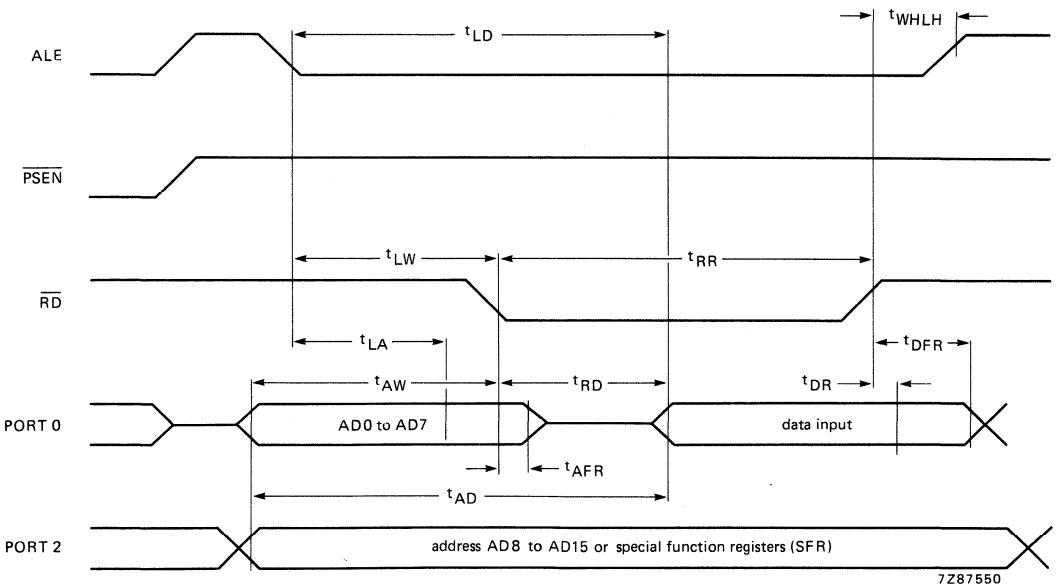


Fig. 13 Read from data memory.

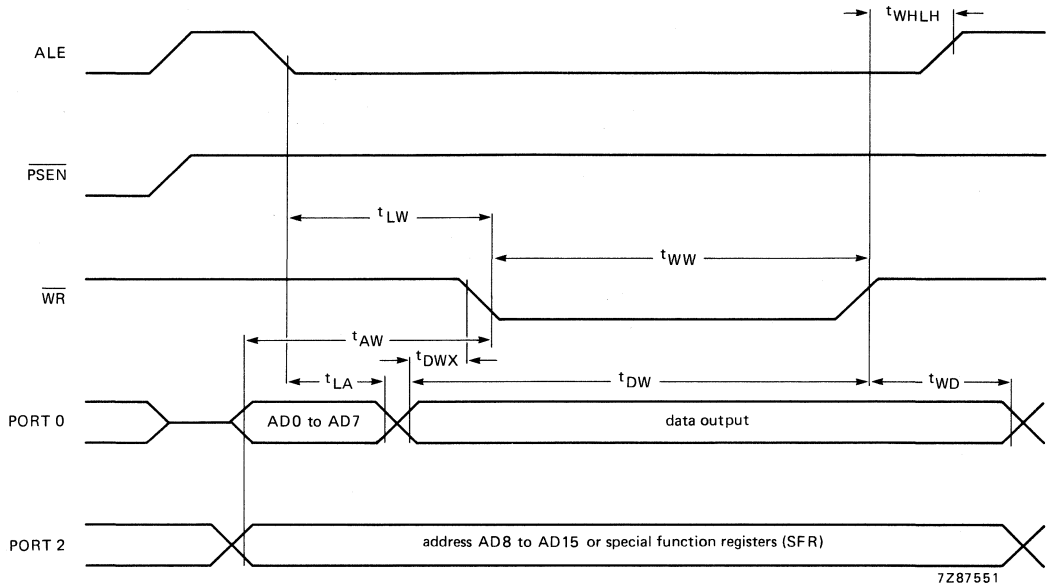


Fig. 14 Write to data memory.

DEVELOPMENT DATA

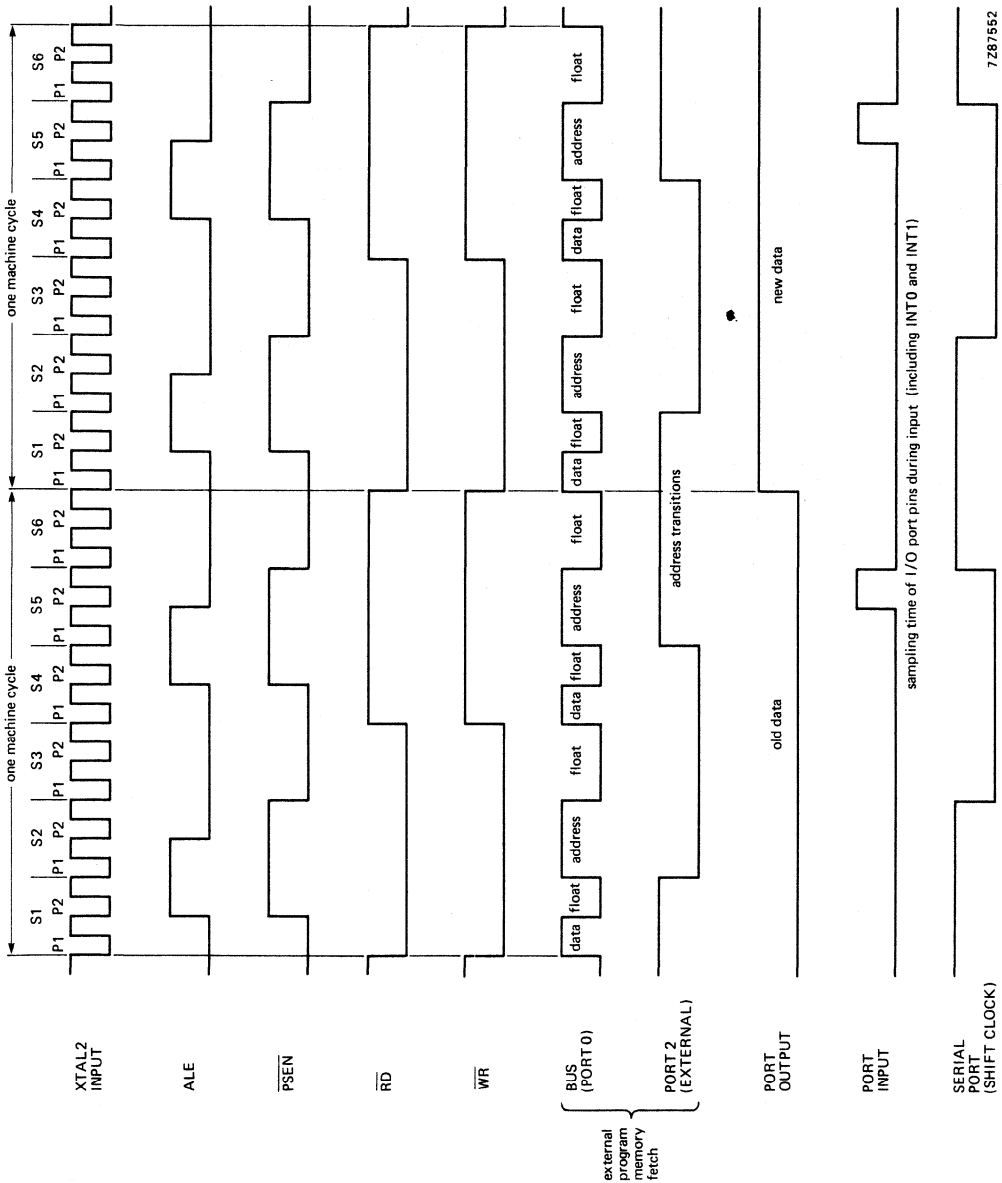


Fig. 15 Instruction cycle timing.

SINGLE-CHIP 8-BIT MICROCONTROLLER

GENERAL DESCRIPTION

The MAB80XXH family of single-chip 8-bit microcontrollers are fabricated in NMOS.
Three interchangeable (pin compatible) versions are available:

- MAB8048H with resident mask-programmed 1 K x 8 ROM, 64 x 8 RAM
- MAB8035HL without resident program memory for use with external EPROM/ROM
- MAB8049H with resident mask-programmed 2 K x 8 ROM, 128 x 8 RAM ←
- MAB8039HL without resident program memory for use with external EPROM/ROM
- MAB8050H with resident mask-programmed 4 K x 8 ROM, 256 x 8 RAM
- MAB8040HL without resident program memory for use with external EPROM/ROM

The MAB80XXH family are designed to be efficient control processors as well as arithmetic processors. Their instruction set allows the user to directly set and reset individual I/O lines as well as test individual bits within the accumulator. A large variety of branch and table look-up instructions enable efficient implementation of standard logic functions. Code efficiency is high; over 70% of the instructions are single byte; all others are two byte.

An on-chip 8-bit counter is provided, which can count either machine cycles (÷ 32) or external events. The counter can be programmed to cause an interrupt to the processor.
Program and data memories plus input/output capabilities can be expanded using standard devices.
For details see users manual 'single-chip microcomputer'.

Features

- 8-bit CPU, ROM, RAM and I/O
- Internal counter/timer
- Internal oscillator, clock driver
- Single-level interrupts: external and counter/timer
- 17 internal registers: accumulator, 16 addressable registers
- Over 90 instructions: 70% single byte
- All instructions 1 or 2 cycles
- Easily expandable memory and 27 I/O lines
- TTL compatible inputs and outputs
- Single 5 V supply
- Standard and extended temperature range (see Table 5)

Applications

- Peripheral interfaces and controllers
- Test and measuring instruments
- Sequencers
- Modems and data enciphering
- Environmental control systems
- Audio/video systems

PACKAGE OUTLINES

All versions : 40-lead DIL; plastic (SOT-129).
MAB8048H/35HLT : 40-lead mini-pack; plastic (VSO-40; SOT-158A).
MAB80XXH/HLWP : 44-lead plastic leaded chip-carrier (PLCC); SOT-187A. ←

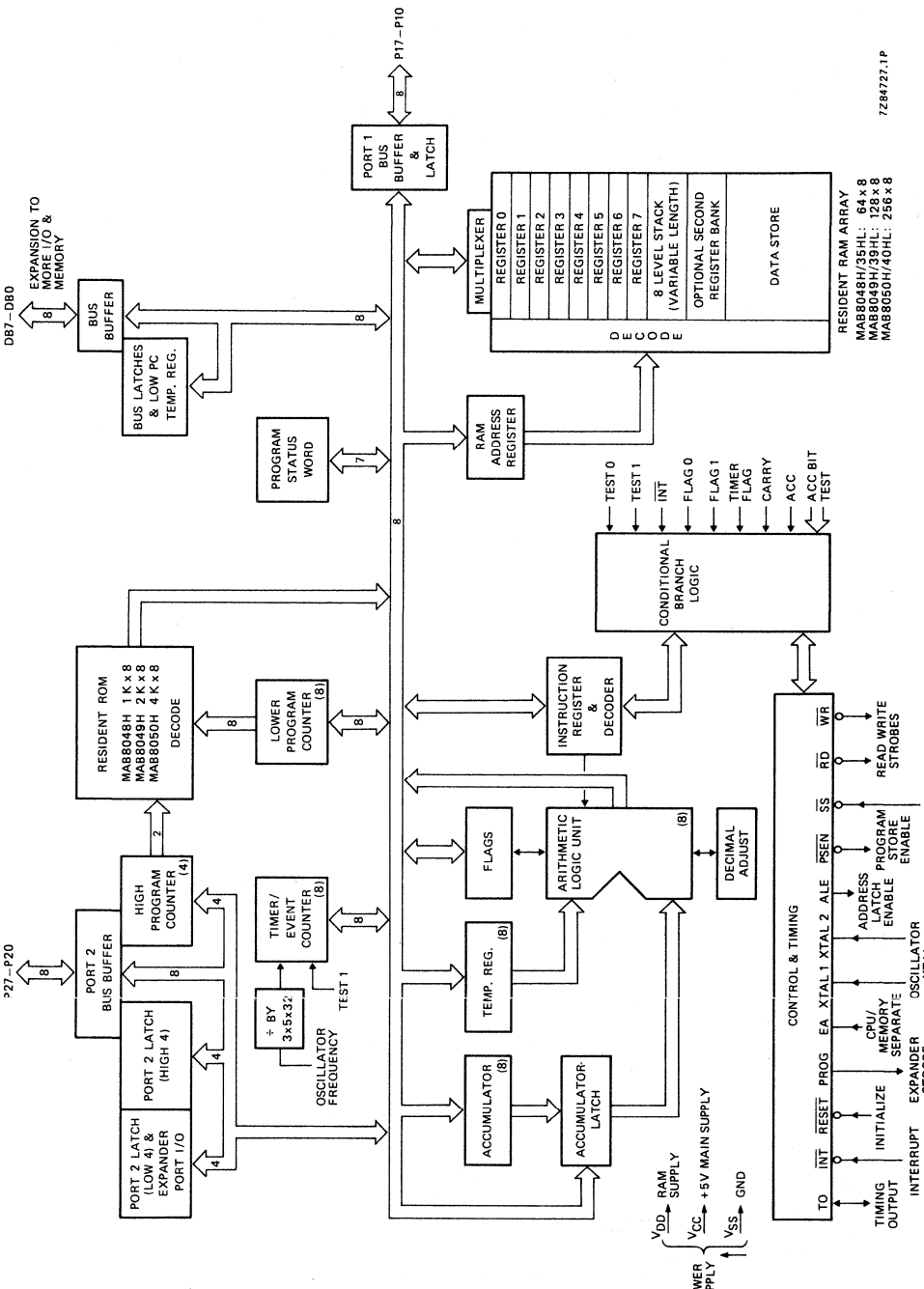


Fig. 1 Block diagram.

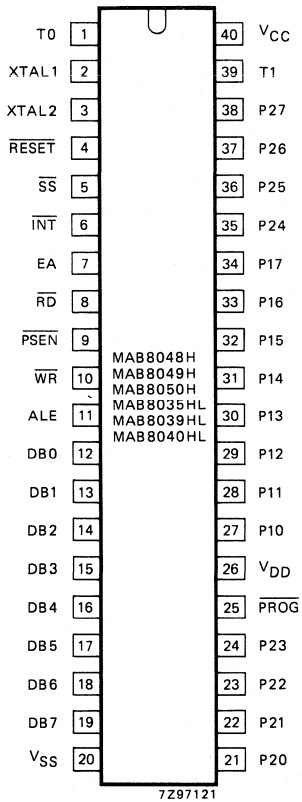
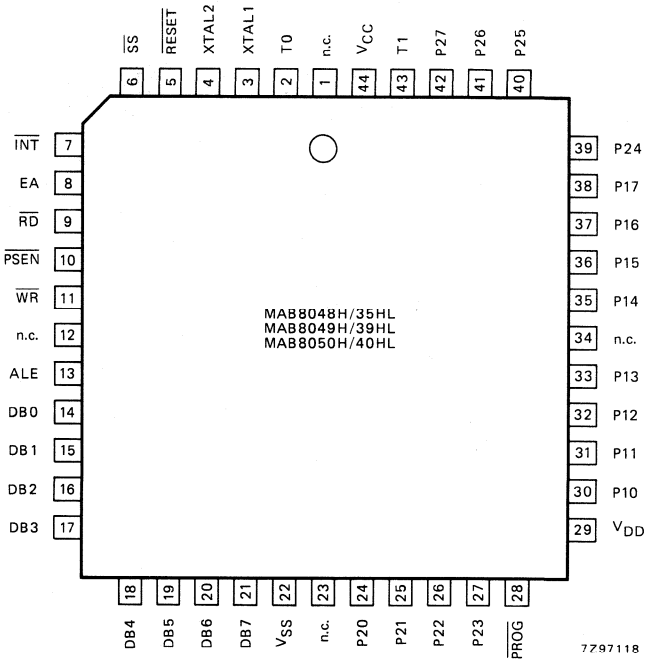


Fig. 2a Pinning diagram; for pin designation see next page.



Where: n.c. = not connected.

Fig. 2b Pinning diagram for MAB80XXH/HLWP; for pin designation see next page.

PINNING

12–19	DB0–DB7	**Data Bus**: bidirectional I/O port which can write or read using the $\overline{RD}$ and $\overline{WR}$ strobes. This port can also be statically latched. It contains the 8 lower order address bits during external memory access and receives the addressed instruction under control of $\overline{PSEN}$. $\overline{PSEN}$, ALE, $\overline{RD}$ and $\overline{WR}$ determine whether the access is an instruction fetch or a read/write access to external RAM.
27–34	P10–P17	**Port 1**: 8-bit quasi-bidirectional I/O port (note 1).
21–24	P10–P27	**Port 2**: 8-bit quasi-bidirectional I/O port (note 1).
35–38		P20–P23 contain the 4 higher order address bits during an access of external program memory.
25	$\overline{PROG}$	**Output strobe**: active LOW for 8243 I/O expander.
1	T0	**Test 0**: test input pin sensed using the JT0 and JNT0 instructions. **Clock**: clock output pin when designated by the ENT0 CLK instructions.
39	T1	**Test 1**: test input pin sensed using the JT1 and JNT1 instructions. Can be designated as the timer/counter input by the STRT CNT instruction.
6	$\overline{INT}$	**Interrupt**: interrupt input pin, which causes an interrupt in the current program, provided that the external interrupt is enabled. Can also be used as an input, testable using the JNI instruction. Interrupt is disabled during and after $\overline{RESET}$.
4	$\overline{RESET}$	**Reset**: active LOW input used to initialize the microcontroller. During program verification the address is latched by a '0' to '1' transition on $\overline{RESET}$ and the data at the addressed location is output on BUS (note 2).
11	ALE	**Address latch enable**: occurs each cycle and is used for timing and sampling. During external program or data memory access, ALE is used to strobe the address information multiplexed on the DB0 to DB7 outputs.
8	$\overline{RD}$	**Read BUS**: active LOW strobe used to gate data onto BUS lines when reading from an external source.
10	$\overline{WR}$	**Write BUS**: active LOW strobe used to write data from BUS lines to an external designation.
7	EA	**External access input**: when HIGH, forces instruction fetch from external memory.
9	$\overline{PSEN}$	**Program store enable**: active LOW strobe that occurs only during a fetch from external memory.
5	$\overline{SS}$	**Single step**: active LOW input used with ALE to cause the microcontroller to execute a single instruction.
2 3	XTAL 1 XTAL 2	**Crystal inputs**: inputs for a crystal, LC-network or an external timing signal to determine the internal oscillator frequency (note 2).
20	V_{SS}	**Ground**: circuit earth potential.
40	V_{CC}	**Power supply**: + 5 V main supply pin.
26	V_{DD}	**Power supply**: + 5 V RAM standby power supply; low power standby pin.

Notes

1. Each port line can be designated as an input or an output. A line is designated as an input by first writing a logic 1 to the line. $\overline{\text{RESET}}$ sets all lines to logic 1.
2. Non-standard TTL V_{IH}.

FUNCTIONAL DESCRIPTION

The following sections provide a detailed functional description of the MAB80XXH microcontroller as shown in Fig. 1. The generic term "MAB80XXH" is used to refer collectively to the MAB8048H/35HL, MAB8049H/39HL and MAB8050H/40HL.

Program memory (see Fig. 3)

The resident program memory consists of a 1024, 2048 or 4096 byte ROM (MAB8048H/49H/50H); the MAB8035HL/39HL/40HL versions do not have a resident program memory. The total addressing capability is 4096 bytes.
The program memory address space is divided into two 2048-byte banks MB0 and MB1. These two 2048 byte banks are divided into 8 pages of 256 bytes for conditional branches.
There are three locations in program memory which contain the first instruction to be executed after one of three events:

- Location 0 – if $\overline{\text{RESET}}$ is activated (LOW) then location 0 is activated
- Location 3 – if the external interrupt is enabled and the $\overline{\text{INT}}$ line is activated (LOW)
- Location 7 – if the T/C interrupt is enabled and the timer/counter has an overflow.

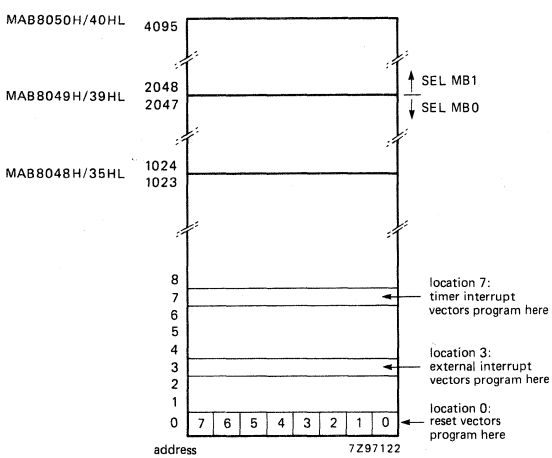


Fig. 3 Program memory map.

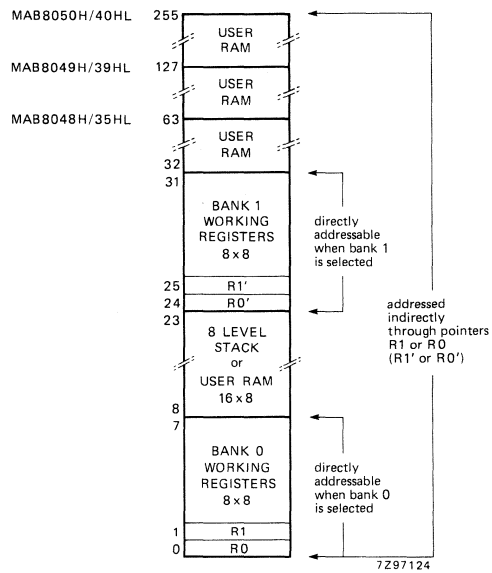


Fig. 4 Data memory map.

FUNCTIONAL DESCRIPTION (continued)

Data memory (see Fig. 4)

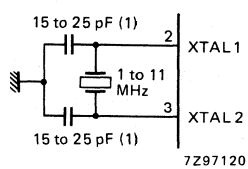
The resident data memory consists of a 64, 128 or 256 byte RAM. All locations are indirectly addressable using two RAM pointer registers R0, R1 or R0', R1'. The first 8 RAM locations (0 to 7) are designated as working registers bank 0 and are directly addressable. By selecting register bank 1, RAM locations 24 to 31 become the working registers, replacing those in register bank 0. RAM locations 8 to 23 are designated as the stack. Two bytes are used per CALL allowing up to 8 levels of subroutine nesting.
An extra 256 bytes of RAM may be added and addressed directly using the MOVX instructions. If the required extra RAM is greater than 320 bytes, additional (256 byte) banks of external memory can be selected, one at a time, by an I/O port.

Program counter and stack

The program counter (PC) is a 12-bit counter/register that points to the location from which the next instruction is to be fetched. When EA is logic 0 the PC can address locations 0 to 1023 (8048H), 2047 (8049H) or 4095 (8050H) of internal program memory. At the 1 K (8048H), 2 K (8049H) boundary, an automatic switch-over to external memory occurs. When EA is logic 1 all the program is fetched from external ROM/EPROM. The total address space is 4 K bytes. An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the program stack. The pair to be used is determined by a 3-bit stack pointer which is part of the program status word (PSW). Data RAM locations 8 to 23 are available as stack registers and are used to store the program counter and 4 bits of PSW. The stack pointer, when initialized to 000B, points to RAM locations 8 and 9. The first subroutine jump or interrupt results in the program counter contents being transferred to locations 8 and 9 of the RAM array. The stack pointer is then incremented by one to point to locations 10 and 11 in anticipation of another CALL. Nesting of subroutines within subroutines can continue up to 8 times without overflowing the stack. If overflow does occur the deepest address stored (locations 8 and 9) will be overwritten and lost since the stack pointer overflows from 111 to 000. It also underflows from 000 to 111. The end of a subroutine, which is signalled by a return instruction (RET or RETR), causes the stack pointer to be decremented and the contents of the resulting register pair to be transferred to the program counter.

Oscillator and clock (see Figs 5, 6 and 7)

The MAB80XXH contains its own internal oscillator and clock driver. A crystal, LC-network or external timing signal (pulse generator) determines the oscillator frequency. The output of the oscillator is divided-by-three and is available at T0 (pin 1) by executing the ENT0 CLK instruction. This clock signal (CLK) is divided-by-five to define a machine (instruction) cycle. It is available at ALE (pin 11).



(1) Including crystal-socket stray capacities.

Fig. 5 Crystal oscillator mode. Crystal series impedance should be $< 75\ \Omega$ at 6 MHz and $< 180\ \Omega$ at 3,6 MHz. When using a ceramic oscillator both capacitors should be 30 pF.

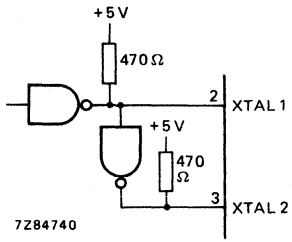
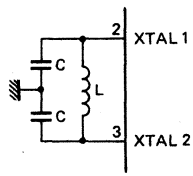


Fig. 6 Driving from external source. Both XTAL 1 and XTAL 2 should be driven. Resistors to V_{CC} (+ 5 V) are required to ensure V_{IH} = 3,8 V if TTL circuitry is used. The minimum HIGH and LOW times are 45%.

L (μH)	C (pF)	nom. f (MHz)
45	20	5,2
120	20	3,2



$$f \approx \frac{1}{2\pi\sqrt{LC'}} \; ; \; C' = \frac{C + 3C_{pp}}{2}$$

Fig. 7 LC oscillator mode. Each C should be ≈ 20 pF including stray capacitance C_{pp} ≈ 5 to 10 pF (pin-to-pin capacitance).

Timer/event counter

An internal counter is available which can count either external events or machine cycles (÷ 32). The machine cycles are divided-by-32 before they are applied to the input of the 8-bit counter. External events are applied directly to the input of the counter. The maximum frequency that can be counted is one third of the machine cycle frequency. The minimum positive duty cycle that can be detected is 0,2 times the cycle period. The counter is under program control and can be made to generate an interrupt to the processor when it overflows.

Interrupt

An interrupt may be generated by:

- An external input $\overline{INT}$ (pin 6)
  or
- Overflow of the internal counter, when enabled.

In either, the processor completes execution of the present instruction followed by a CALL to the interrupt service routine.
After service, a RETR instruction restores the machine to the state it was prior to the interrupt. The external interrupt has priority over the internal interrupt.

Input/output

The MAB80XXH has 27 I/O lines. They are arranged as three 8-line ports which serve as either inputs, outputs or bidirectional ports and 3 'test' inputs that can alter program sequences when tested by conditional jump instructions.

FUNCTIONAL DESCRIPTION (continued)

Ports 1 and 2 (see Fig. 8)

Ports 1 and 2 are both 8-bits wide and have identical characteristics. Data written to these ports is statically latched and remains unchanged until rewritten. As input lines these ports are non-latching, i.e. inputs must be present until read by an input instruction. Inputs are fully TTL compatible and outputs will drive one standard TTL load.

The lines of ports 1 and 2 are called quasi-bidirectional because of a special output circuit structure which allows each line to serve as an input, an output, or both even through outputs are statically latched. Each line is continuously pulled up to + 5 V through a resistor (≈ 50 Ω). Thus pull-up provides sufficient source current for a TTL HIGH level, yet can be pulled LOW by a standard TTL gate, thus allowing the pin to be used both as an input and an output. To provide fast switching times during a logic 0 to 1 transition, transistor TR 2 is switched on for one fifth of a machine cycle when a logic 1 is written to the line. When a logic 0 is written transistor TR 1 overcomes the pull-up and provides TTL current sinking capability. Since the pull-down transistor is low impedance, a logic 1 must first be written to any line which is to used as an input. $\overline{\text{RESET}}$ initializes all lines to the high impedance logic 1 state. This structure allows input and output on the same pin plus a mixture of input and output lines on the same port. The quasi-bidirectional port in combination with the ANL and ORL logical instructions facilitates the handling of single line inputs and outputs within an 8-bit microcontroller.

BUS (DB0–DB7)

Bus is a true bidirectional 8-bit port with associated input and output strobes. If the bidirectional feature is not required, BUS can serve as a statically latched output port or non-latching input port. Input and output lines on this port cannot be mixed.

As a static port, data is written and latched using the OUTL instruction and input using the INS instruction. These instructions generate pulses on the corresponding $\overline{\text{RD}}$ and $\overline{\text{WR}}$ output strobe lines, but in the static port mode they are not used. As a bidirectional port, the MOVX instructions are used to read from and write to the port. A write generates a pulse on the $\overline{\text{WR}}$ output line and output data is valid at the trailing edge of $\overline{\text{WR}}$. A read generates a pulse on the $\overline{\text{RD}}$ output line and input data must be valid at the trailing edge of $\overline{\text{RD}}$. When the BUS lines are not being written to or read from they are high impedance.

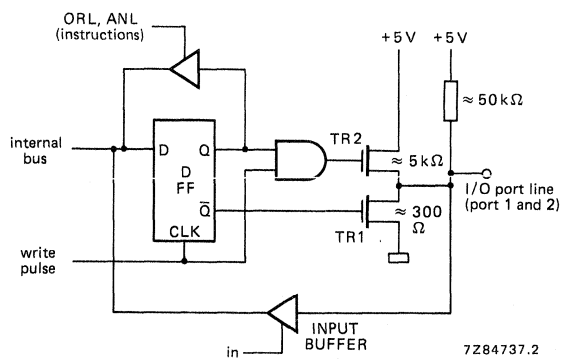


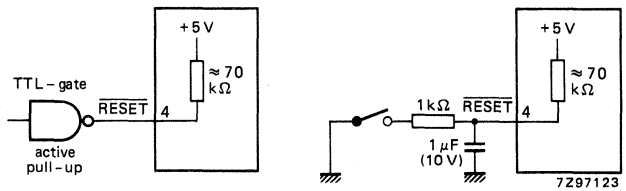
Fig. 8 Quasi-bidirectional port structure.

Test (T0, T1) and $\overline{INT}$

These three pins serve as inputs and are testable with the conditional jump instruction. They allow inputs to cause program branches without the necessity of loading an input port into the accumulator.

$\overline{RESET}$ (see Fig. 9)

This active LOW input is used to initialize the microcontroller.
This Schmitt-trigger input has an internal pull-up resistor which, in combination with an internal 1 μF capacitor, provides an internal reset pulse of sufficient duration to reset all circuitry. If the reset pulse is generated externally, the reset pin must be held at ground (0,45 V) for at least 10 ms after the power supply is within tolerance. Only 5 machine cycles (12,5 μs at 6 MHz) are required if power is already on and the oscillator has stabilized.



(a) (b)

Fig. 9 An external reset is shown in (a) and power-on reset in (b).

Single step ($\overline{SS}$)

This active LOW input when used in combination with ALE will cause the microcontroller to execute a single instruction, then wait until $\overline{SS}$ is reactivated.

Power-down mode (see Fig. 10)

In the MAB80XXH, power can be removed from all but the data RAM array, for low power standby operation. In the power-down mode the contents of the data RAM can be maintained while drawing typically 10% to 15% of the normal operating supply voltage. V_{CC} serves as the + 5 V supply pin for the bulk of the circuitry, while the V_{DD} pin supplies only the RAM array. In normal operation, both pins are at + 5 V. In the standby mode, V_{CC} is at ground and only V_{DD} is maintained at + 5 V. Applying $\overline{RESET}$ to the microcontroller through the reset pin inhibits any access to the RAM and ensures that the RAM cannot be inadvertently altered as power is removed from V_{CC}.

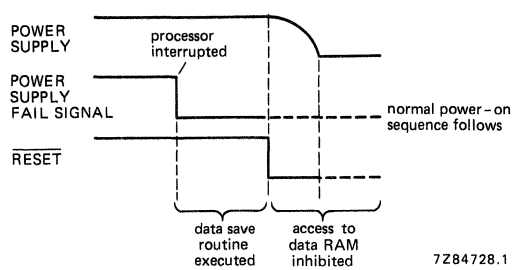


Fig. 10 Power down sequence.

FUNCTIONAL DESCRIPTION (continued)

Instruction set (see Tables 1, 2, 3 and 4)

The MAB80XXH instruction set consists of over 90 one and two-byte instructions. Program code efficiency is high because:

- Working registers and program variables are stored in the RAM locations 0 to 127, which require only a single byte to address
- Program memory is divided into pages of 256 bytes, which means that branch destination addresses require only one byte

The instruction set performs logical, arithmetic and test operations on bytes. It also manipulates and tests bits. A set of MOVE instructions operate indirectly on either RAM or ROM, which permits efficient access of pointers and data tables. The indirect jump instruction performs a multi-way branch (up to 256) on the contents of the accumulator to addresses stored in a look-up table. The 'decrement register and jump if not zero' instruction saves a bit each time it is used as opposed to using separate increment and test instructions. The on-chip counter provides the facility for external events or time to be counted off-line from the main program. The MAB80XXH can either test the counter (under program control) or cause its overflow to generate an interrupt. These features are essential for real-time applications.

Table 1 Symbols and definitions used in Table 2.

symbol	definition description
A	accumulator
addr	program memory address
Bb	bit designation (b = 0–7)
RBS	register bank select
C	carry (bit CY)
CNT	event counter
D	mnemonic for 4-bit digit (nibble)
data	8-bit number or expression
I	interrupt
MB	memory bank
MBFF	memory bank flip-flop
P	mnemonic for 'in-page' operation
PC	program counter
Pp	port designation (p = 0, 1, 2)
PSW	program status word
RB	register bank
Rr	register designation (r = 0–7)
Sn	serial I/O register
SP	stack pointer
T	timer
TF	timer flag
T0	test 0 input
T1	test 1 input
#	immediate data prefix
@	indirect address prefix
(X)	contents of X
((X))	contents of location addressed by X
←	is replaced by
↔	is exchanged with

Table 2 Instruction set

	mnemonic	opcode (hex.)	bytes/ cycles	description	function		notes
ACCUMULATOR	ADD A, Rr	6*	1/1	Add register contents to A	(A)←(A) + (Rr)	r = 0–7	1
	ADD A, @Rr	60 61	1/1	Add RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) (A)←(A) + ((R1))		1
	ADD A, #data	03 data	2/2	Add immediate data to A	(A)←(A) + data		1
	ADDC A, Rr	7*	1/1	Add carry and register contents to A	(A)←(A) + (Rr) + (C)	r = 0–7	1
	ADDC A, @Rr	70 71	1/1	Add carry and RAM data, addresses by Rr, to A	(A)←(A) + ((R0)) + (C) (A)←(A) + ((R1)) + (C)		1
	ADDC A, #data	13 data	2/2	Add carry and immediate data to A	(A)←(A) + data + (C)		1
	ANL A, Rr	5*	1/1	'AND' Rr with A	(A)←(A) AND (Rr)	r = 0–7	
	ANL A, @Rr	50 51	1/1	'AND' RAM data, addressed by Rr, with A	(A)←(A) AND ((R0)) (A)←(A) AND ((R1))		
	ANL A, #data	53 data	2/2	'AND' immediate data with A	(A)←(A) AND data		
	ORL A, Rr	4*	1/1	'OR' Rr with A	(A)←(A) OR (Rr)	r = 0–7	
	ORL A, @Rr	40 41	1/1	'OR' RAM data, addressed by Rr, with A	(A)←(A) OR ((R0)) (A)←(A) OR ((R1))		
	ORL A, #data	43 data	2/2	'OR' immediate data with A	(A)←(A) OR data		
	XRL A, Rr	D*	1/1	'XOR' Rr with A	(A)←(A) XOR (Rr)	r = 0–7	
	XRL A, @Rr	D0 D1	1/1	'XOR' RAM, addressed by Rr, with A	(A)←(A) XOR ((R0)) (A)←(A) XOR ((R1))		
	XRL A, #data	D3 data	2/2	'XOR' immediate data with A	(A)←(A) XOR data		
	INC A	17	1/1	increment A by 1	(A)←(A) + 1		
	DEC A	07	1/1	decrement A by 1	(A)←(A) – 1		
	CLR A	27	1/1	clear A to zero	(A)←0		
	CPL A	37	1/1	one's complement A	(A)←NOT(A)		
	RL A	E7	1/1	rotate A left	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(A_7)$	n = 0–6	

	mnemonic	opcode (hex.)	bytes/ cycles	description	function		notes
ACCUMULATOR (cont.)	RLC A	F7	1/1	rotate A left through carry	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$	n = 0–6	2
	RR A	77	1/1	rotate A right	$(A_n)\leftarrow(A_{n+1})$ $(A_7)\leftarrow(A_0)$	n = 0–6	
	RRC A	67	1/1	rotate A right through carry	$(A_n)\leftarrow(A_{n+1})$ $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$	n = 0–6	2
	DA A	57	1/1	decimal adjust A			2
	SWAP A	47	1/1	swap nibbles of A	$(A_{4-7})\leftrightarrow(A_{0-3})$		
DATA MOVES	MOV A, Rr	F*	1/1	move register contents to A	(A)←(Rr)	r = 0–7	
	MOV A, @Rr	F0 F1	1/1	move RAM data, addressed by Rr, to A	(A)←((R0)) (A)←((R1))		
	MOV A, #data	23 data	2/2	move immediate data to A	(A)←data		
	MOV Rr, A	A*	1/1	move accumulator contents to register	(Rr)←(A)	r = 0–7	
	MOV @Rr, A	A0 A1	1/1	move accumulator contents to RAM location addressed by Rr	((R0))←(A) ((R1))←(A)		
	MOV Rr, #data	B* data	2/2	move immediate data to Rr	(Rr)←data	r = 0–7	
	MOV @rR, #data	B0 data B1 data	2/2	move immediate data to RAM location addressed by Rr	((R0))←data ((R1))←data		
	XCH A, Rr	2*	1/1	exchange accumulator contents with Rr	(A)↔(Rr)	r = 0–7	
	XCH A, @Rr	20 21	1/1	exchange accumulator contents with RAM data addressed by Rr	(A)↔((R0)) (A)↔((R1))		
	XCHD A, @Rr	30 31	1/1	exchange lower nibbles of A and RAM data addressed by Rr	$(A_{0-3})\leftrightarrow((R0_{0-3}))$ $(A_{0-3})\leftrightarrow((R1_{0-3}))$		
	MOV A, PSW	C7	1/1	move PSW contents to accumulator	(A)←(PSW)		
	MOV PSW, A	D7	1/1	move accumulator contents to PSW	(PSW)←(A)		3
	MOVP A, @A	A3	1/2	move indirectly addressed data in current page to A	(A)←((A))		
	MOVP3 A,@A	E3	1/2	move data in page 3 to A	(A)←((A))	in page 3	

	MOVX A,@Rr	80 81	1/2	move indirect the contents of external memory to A	(A)←((R))	r = 0–1	
	MOVX @Rr,A	90 91	1/2	move indirect the contents of A to external memory	((R))←(A)	r = 0–1	
FLAGS	CLR C	97	1/1	clear carry bit	(C)←0		2
FLAGS	CPL C	A7	1/1	complement carry bit	(C)←NOT(C)		2
REGISTER	INC Rr	1*	1/1	increment register by 1	(Rr)←(Rr) + 1	r = 0–7	
REGISTER	INC @Rr	10 11	1/1	increment RAM data, addressed by Rr, by 1	((R0))←((R0)) + 1 ((R1))←((R1)) + 1		
REGISTER	DEC Rr	C*	1/1	decrement register by 1	(Rr)←(Rr) –1	r = 0–7	
BRANCH	JMP addr	● 4 address	2/2	unconditional jump within a 2 K bank	(PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1		
BRANCH	JMPP @A	B3	1/2	indirect jump within a page	(PC_{0-7})←((A))		
BRANCH	DJNZ Rr, addr	E* address	2/2	decrement Rr by 1 and jump if not zero to addr	(Rr)←(Rr) –1 if (Rr) not zero (PC_{0-7})←addr	r = 0–7	
BRANCH	JF0 addr	B6 address	2/2	jump to addr if F0 = 1	if F0 = 1: (PC_{0-7})←addr		
BRANCH	JF1 addr	76 address	2/2	jump to addr if F1 = 1	if F1 = 1: (PC_{0-7})←addr		
BRANCH	JNI addr	86 address	2/2	jump to addr if $\overline{INT}$ = 0	if $\overline{INT}$ = 0: (PC_{0-7})←addr		
BRANCH	JBb addr	▲ 2 address	2/2	jump to addr if Acc. bit b = 1	if b = 1: (PC_{0-7})←addr	b = 0–7	
BRANCH	JC addr	F6 address	2/2	jump to addr if C = 1	if C = 1: (PC_{0-7})←addr		
BRANCH	JNC addr	E6 address	2/2	jump to addr if C = 0	if C = 0: (PC_{0-7})←addr		
BRANCH	JZ addr	C6 address	2/2	jump to addr if A = 0	if A = 0: (PC_{0-7})←addr		
BRANCH	JNZ addr	96 address	2/2	jump to addr if A is NOT zero	if A ≠0: (PC_{0-7})←addr		
BRANCH	JT0 addr	36 address	2/2	jump to addr if T0 = 1	if T0 = 1: (PC_{0-7})←addr		
BRANCH	JNT0 addr	26 address	2/2	jump to addr if T0 = 0	if T0 = 0: (PC_{0-7})←addr		
BRANCH	JT1 addr	56 address	2/2	jump to addr if T1 = 1	if T1 = 1: (PC_{0-7})←addr		
BRANCH	JNT1 addr	46 address	2/2	jump to addr if T1 = 0	if T1 = 0: (PC_{0-7})←addr		
BRANCH	JTF addr	16 address	2/2	jump to addr if Timer Flag = 1	if TF = 1: (PC_{0-7})←addr		4

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
TIMER/EVENT COUNTER	MOV A, T	42	1/1	move timer/event counter contents to accumulator	(A)←(T)	
	MOV T, A	62	1/1	move accumulator contents to timer/event counter	(T)←(A)	
	STRT CNT	45	1/1	start event counter		
	STRT T	55	1/1	start timer		
	STOP TCNT	65	1/1	stop timer/event counter		
	EN TCNTI	25	1/1	enable timer/event counter interrupt		
	DIS TCNTI	35	1/1	disable timer/event counter interrupt		
CONTROL	EN I	05	1/1	enable external interrupt		
	DIS I	15	1/1	disable external interrupt		
	SEL RB0	C5	1/1	select register bank 0	(RBS)←0	5
	SEL RB1	D5	1/1	select register bank 1	(RBS)←1	5
	SEL MB0	E5	1/1	select program memory bank 0	(MBFF0)←0, (MBFF1)←0	
	SEL MB1	F5	1/1	select program memory bank 1	(MBFF0)←1, (MBFF1)←0	
	ENT0 CLK	75	1/1	enable clock output onto T0		
SUBROUTINE	CALL addr	▲ 4 address	2/2	jump to subroutine	((SP))←(PC), ($PSW_{4, 6, 7}$) (SP)←(SP) + 1 (PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1	6
	RET	83	1/2	return from subroutine	(SP)←(SP) −1 (PC)←((SP))	6
	RETR	93	1/2	return from interrupt and restore bits 4, 6, 7 of PSW	(SP)←(SP) −1 ($PSW_{4, 6, 7}$) + (PC)←((SP))	6

INPUT/OUTPUT	IN A,Pp	09 0A	1/2	input port p data to accumulator	(A)←(P1) (A)←(P2)	p = 1–2	7
	INS A,BUS	08	2/1	input strobed BUS data into accumulator	(A)←(BUS)		
	OUTL Pp,A	39 3A	1/2	output accumulator data to port p	(P1)←(A) (P2)←(A)	p = 1–2	
	ANL BUS, #data	98	2/2	logical AND immediate data with BUS	(BUS)←(BUS) AND data		
	ANL Pp, #data	99 9A	2/2	AND port p data with immediate data	(P1)←(P1) AND data (P2)←(P2) AND data	p = 1–2	
	ORL Pp, #data	89 8A	2/2	OR port p data with immediate data	(P1)←(P1) OR data (P2)←(P2) OR data	p = 1–2	
	ORL BUS, #data	88	2/2	logical OR immediate data with BUS	(BUS)←(BUS) OR data		
	MOVD A,Pp	0C 0D 0E 0F	1/2	move contents of designated port (4–7) to A	(A_{0-3})←(Pp) (A_{4-7})←0	p = 4–7	
	MOVD Pp,A	3C 3D 3E 3F	1/1	move contents of A to designated port (4–7)	(Pp)←(A_{0-3})	p = 4–7	
	ANLD Pp,A	9C 9D 9E 9F	1/2	logical AND contents of A with designated port (4–7)	(Pp)←(Pp) AND (A_{0-3}	p = 4–7	
	ORLD Pp,A	8C 8D 8E 8F	1/1	logical OR contents of A with designated port (4–7)	(Pp)←(Pp) OR (A_{0-3})	p = 4–7	
	NOP	00	1/1	no operation			

Notes to Table 2.

1. **PSW** CY, AC affected
2. **PSW** CY affected
3. PSW PS affected
4. Execution of JTF and JNTF instructions resets the Timer Flag (TF).
5. PSW RBS affected
6. PSW SP_0, SP_1, SP_2 affected
7. (A) = 111 P23, P22, P21, P20.
8. (S1) has a different meaning for read and write operation, see serial I/O interface.

* : 8, 9, A, B, C, D, E, F
● : 0, 2, 4, 6, 8, A, C, E
▲ : 1, 3, 5, 7, 9, B, D, F

Table 3 Instruction timing (see also Figs 11 and 12)

	cycle 1					cycle 2				
instruction	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,P	fetch instruction	increment program counter	–	increment timer	–	–	read port	* –	–	–
OUTL P,A			–		output to port	–	–	* –	–	–
ANL P,#data		*	–		read port	fetch immediate data	–	*increment program counter	output to port	–
ORL P,#data		*	–		read port	fetch immediate data	–	*increment program counter	output to port	–
INS A,BUS			–		–	–	read port	* –	–	–
OUTL BUS,A			–		output to port	–	–	* –	–	–
ANL BUS,#data		*	–		read port	fetch immediate data	–	*increment program counter	output to port	–
ORL BUS,#data		*	–		read port	fetch immediate data	–	*increment program counter	output to port	–
MOVX @R,A			output RAM address		output data to RAM	–	–	* –	–	–
MOVX A,@R			output RAM address		–	–	read data	* –	–	–
MOVD A,P	fetch instruction	increment program counter	output opcode/ address	increment timer	–	–	read P2 lower	* –	–	–

	cycle 1					cycle 2				
instruction	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
MOVD P,A	fetch instruction	increment program counter	output opcode/ address	increment timer	output data to P2 lower	–	–	* –	–	–
ANLD P,A	¦		output opcode/ address	¦	output data	–	–	* –	–	–
ORLD P,A	¦		output opcode/ address	¦	output data	–	–	* –	–	–
J (conditional)	¦	*	sample condition	increment timer	–	fetch immediate data	–	* opdata program counter	–	–
STRT CNT STRT T	¦	*	–	–	start counter					
STOP TCNT	¦	*	–	–	stop counter					
EN I	¦	*	–	enable interrupt	–					
DIS I	¦	*	–	disable interrupt	–					
ENTO CLK	fetch instruction	* increment program counter	–	enable clock	–					

* Valid instruction addresses are output at this time if external program memory is being accessed.

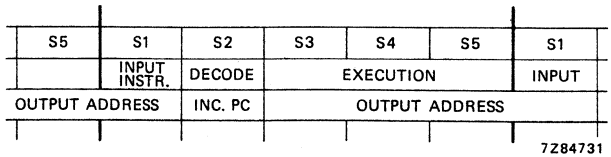


Fig. 11 Instruction cycle.

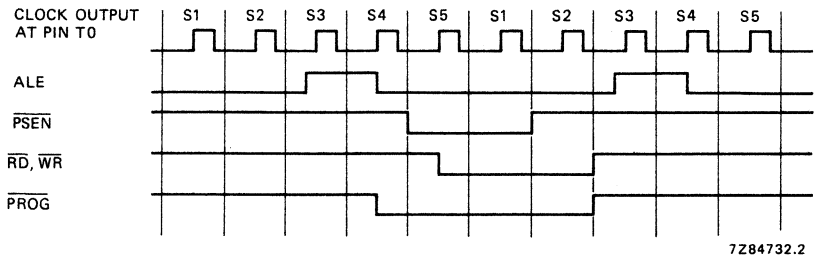


Fig. 12 Instruction cycle timing.

Table 4 Instruction map.

first hexadicimal character of opcode ↓ / second hexadicimal character of opcode →	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP		OUTL BUS,A	ADD A, #data	JMP page 0	EN I		DEC A	INS A,BUS	IN A, Pp	2		MOVD A,Pp	4	5	6	7
1	INC @Rr		JB0 addr	ADDC A, #data	CALL page 0	DIS I	JTF addr	INC A	INC Rr								
	0	1							0	1	2	3	4	5	6	7
2	XCH A, @Rr			MOV A, #data	JMP page 1	EN TCNTI	JNT0 addr	CLR A	XCH A,Rr								
	0	1							0	1	2	3	4	5	6	7
3	XCHD A, @Rr		JB1 addr		CALL page 1	DIS TCNTI	JT0 addr	CPL A		OUTL Pp,A			MOVD Pp,A				
	0	1								1	2		4	5	6	7
4	ORL A, @Rr		MOV A, T	ORL A, #data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	ORL A,Rr								
	0	1							0	1	2	3	4	5	6	7
5	ANL A, @Rr		JB2 addr	ANL A, #data	CALL page 2	STRT T	JT1 addr	DA, A	ANL A,Rr								
	0	1							0	1	2	3	4	5	6	7
6	ADD A, @Rr		MOV T,A		JMP page 3	STOP TCNT		RRC A	ADD A,Rr								
	0	1							0	1	2	3	4	5	6	7
7	ADDC A, @Rr		JB3 addr		CALL page 3	ENT0 CLK	JF1 addr	RR A	ADDC A,Rr								
	0	1							0	1	2	3	4	5	6	7
8	MOVX A, @Rr			RET	JMP page 4	CLR F0	JN1 addr		ORL BUS, #data	ORL Pp, #data			ORLD Pp,A				
	0	1								1	2		4	5	6	7
9	MOVX @Rr,A		JB4 addr	RETR	CALL page 4	CPL F0	JNZ addr	CLR C	ANL BUS, #data	ANP Pp, #data			ANLD Pp,A				
	0	1								1	2		4	5	6	7
A	MOV @Rr, A			MOVP A, @A	JMP page 5	CLR F1		CPL C	MOV Rr,A								
	0	1							0	1	2	3	4	5	6	7
B	MOV @Rr, #data		JB5 addr	JMPP @A	CALL page 5	CPL F1	JF0 addr		MOV R, #data								
	0	1							0	1	2	3	4	5	6	7
C					JMP page 6	SEL RB0	JZ addr	MOV A, PSW	DEC Rr								
									0	1	2	3	4	5	6	7
D	XRL A, @Rr		JB6 addr	XRL A, #data	CALL page 6	SEL RB1		MOV PSW, A	XRL A,Rr								
	0	1							0	1	2	3	4	5	6	7
E				MOVP3 A @A	JMP page 7	SEL MB0	JNC addr	RL A	DJNZ Rr, addr								
									0	1	2	3	4	5	6	7
F	MOV A, @Rr		JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	MOV A,Rr								
	0	1							0	1	2	3	4	5	6	7

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Input voltage with respect to V_{SS}			
except input EA	V_I	−0,5 to + 7	V
input EA	V_I	−0,5 to + 12	V
D.C. current into any input or output	$\pm I_I, \pm I_O$	max. 10	mA
Total power dissipation	P_{tot}	max. 1	W
Storage temperature range	T_{stg}	−65 to +150	°C
Operating ambient temperature range	T_{amb}	see Table 5	

Table 5 MAB80XXH versions

version	internal memory		RAM st/by.	frequency (MHz) min.	frequency (MHz) max.	temperature range (°C)
MAB8048H	1 K x 8 ROM	64 byte RAM	yes	1,0	11,0	0 to + 70
MAB8035HL	none	64 byte RAM	yes	1,0	11,0	0 to + 70
MAF8048H	1 K x 8 ROM	64 byte RAM	yes	1,0	11,0	−40 to + 85
MAF8035HL	none	64 byte RAM	yes	1,0	11,0	−40 to + 85
MAF80A48H	1 K x 8 ROM	64 byte RAM	yes	1,0	10,0	−40 to + 110
MAF80A35HL	none	64 byte RAM	yes	1,0	10,0	−40 to + 110
MAB8049H	2 K x 8 ROM	128 byte RAM	yes	1,0	11,0	0 to + 70
→ MAB8039HL	none	128 byte RAM	yes	1,0	11,0	0 to + 70
MAF8049H	2 K x 8 ROM	128 byte RAM	yes	1,0	11,0	−40 to + 85
MAF8039HL	none	128 byte RAM	yes	1,0	11,0	−40 to + 85
MAF80A49H	2 K x 8 ROM	128 byte RAM	yes	1,0	10,0	−40 to + 110
MAF80A39HL	none	128 byte RAM	yes	1,0	10,0	−40 to + 110
MAB8050H	4 K x 8 ROM	256 byte RAM	yes	1,0	6,0	0 to + 70
→ MAB8040HL	none	256 byte RAM	yes	1,0	6,0	0 to + 70
MAF8050H	4 K x 8 ROM	256 byte RAM	yes	1,0	6,0	−40 to + 85
→ MAF8040HL	none	256 byte RAM	yes	1,0	6,0	−40 to + 85
MAF80A50H	4 K x 8 ROM	256 byte RAM	yes	1,0	6,0	−40 to + 110
MAF80A40HL	none	256 byte RAM	yes	1,0	6,0	−40 to + 110

D.C. CHARACTERISTICS (MAB8048H/35HL; MAB8049H/39HL; MAB8050H/40HL)

$V_{CC} = V_{DD} = 5$ V (± 10%); $V_{SS} = 0$ V; $T_{amb} = 0$ to + 70 °C; all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	typ.	max.	unit
Supply current at $V_{DD} = 5$ V ± 10%; $V_{SS} = V_{CC} = 0$ V					
MAB8048H/35HL	I_{DD}	–	–	6	mA
MAB8049H/39HL	I_{DD}	–	–	8	mA
MAB8050H/40HL	I_{DD}	–	–	15	mA
Supply current (total) at $V_{DD} = 5$ V ± 10%; $V_{SS} = V_{CC} = 0$ V					
MAB8048H/35HL	$I_{DD} + I_{CC}$	–	–	80	mA
MAB8049H/39HL	$I_{DD} + I_{CC}$	–	–	90	mA
MAB8050H/40HL	$I_{DD} + I_{CC}$	–	–	100	mA
Inputs					
Input voltage LOW all inputs except XTAL 1, XTAL 2, $\overline{\text{RESET}}$	V_{IL}	−0,5	–	0,8	V
Input voltage LOW XTAL 1, XTAL 2, $\overline{\text{RESET}}$	V_{IL1}	−0,5	–	0,6	V
Input voltage HIGH all inputs except XTAL 1, XTAL 2, $\overline{\text{RESET}}$	V_{IH}	2,0	–	V_{CC}	V
Input voltage HIGH XTAL 1, XTAL 2, $\overline{\text{RESET}}$	V_{IH1}	3,8	–	V_{CC}	V
Outputs					
Output voltage LOW (DB0 to DB7) at $I_{OL} = 2$ mA	V_{OL}	–	–	0,45	V
Output voltage LOW ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{PSEN}}$, ALE) at $I_{OL1} = 1{,}8$ mA	V_{OL1}	–	–	0,45	V
Output voltage LOW ($\overline{\text{PROG}}$) at $I_{OL2} = 1$ mA	V_{OL2}	–	–	0,45	V
Output voltage LOW (all other outputs) at $I_{OL3} = 1{,}6$ mA	V_{OL3}	–	–	0,45	V
Output voltage HIGH (DB0 to DB7) at $-I_{OH} = 400\ \mu$A	V_{OH}	2,4	–	–	V
Output voltage HIGH ($\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{PSEN}}$, ALE) at $-I_{OH1} = 100\ \mu$A	V_{OH1}	2,4	–	–	V
Output voltage HIGH (all other outputs) at $-I_{OH} = 40\ \mu$A	V_{OL2}	2,4	–	–	V

D.C. CHARACTERISTICS (continued)

parameter	symbol	min.	typ.	max.	unit
Input leakage current (T1, $\overline{INT}$) at $V_{SS} < V_I < V_{CC}$	$\pm I_{IL}$	–	–	10	μA
Output leakage current (DB0 to DB7; high impedance) at $V_{SS} + 0{,}45\ V < V_I < V_{CC}$	$\pm I_{OZ}$	–	–	10	μA
LOW input load current (P10 to P17, P20 to P27 EA, $\overline{SS}$) at $V_{SS} + 0{,}45\ V < V_I < V_{CC}$	I_{LI}	–	–	−0,5	mA
LOW input load current ($\overline{RESET}$) at $V_{SS} < V_I < V_{CC}$	I_{LI1}	0,02	–	−0,3	mA

D.C. CHARACTERISTICS

MAF8048H/35HL; MAF8049H/39HL; MAF8050H/40HL (at $T_{amb} = -40$ to + 85 °C)
MAF80A48H/A35HL; MAF80A49H/A39HL; MAF80A50H/A40HL (at $T_{amb} = -40$ to + 110 °C)

$V_{CC} = V_{DD} = 5\ V$ (± 10%); $V_{SS} = 0\ V$; T_{amb} as above; all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	typ.	max.	unit
Supply current at $V_{DD} = 5\ V \pm 10\%$; $V_{SS} = V_{CC} = 0\ V$					
MAF8048H/35HL; MAF80A48H/A35HL	I_{DD}	–	–	8	mA
MAF8049H/39HL; MAF80A49H/A39HL	I_{DD}	–	–	10	mA
MAF8050H/40HL; MAF80A50H/A40HL	I_{DD}	–	–	18	mA
Supply current (total) at $V_{DD} = 5\ V \pm 10\%$; $V_{SS} = V_{CC} = 0\ V$					
MAF8048H/35HL; MAF80A48H/A35HL	$I_{DD} + I_{CC}$	–	–	90	mA
MAF8049H/39HL; MAF80A49H/A39HL	$I_{DD} + I_{CC}$	–	–	100	mA
MAF8050H/40HL; MAF80A50H/A40HL	$I_{DD} + I_{CC}$	–	–	120	mA
Inputs					
Input voltage HIGH all inputs except XTAL 1, XTAL 2, $\overline{RESET}$	V_{IH}	2,2	–	V_{CC}	V
LOW input load current (P10 to P17, P20 to P27 EA, $\overline{SS}$) at $V_{SS} + 0{,}45\ V < V_I < V_{CC}$	$-I_{LI}$	–	–	0,6	mA

A.C. CHARACTERISTICS (MAB8048H/35HL; MAB8049H/39HL; MAB8050H/40HL)

$V_{CC} = V_{DD} = 5$ V (± 10%); $V_{SS} = 0$ V; $T_{amb} = 0$ to + 70 °C; note 1.
See waveforms Figs 14, 15, 16, 17 and 18

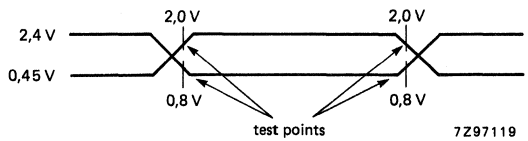
parameter	symbol	f(t_{CY}) (note 2)	11 MHz		unit
			min.	max.	
Clock period (note 2)	t_{CY}	$1/(f_{XTAL})$	90,8	1000	ns
ALE pulse duration	t_{LL}	$3{,}5t_{CY}-170$	150	–	ns
Address set-up time to ALE (note 3)	t_{AL}	$2t_{CY}-110$	70	–	ns
Address hold time after ALE	t_{LA}	$t_{CY}-40$	50	–	ns
Control pulse duration $\overline{RD}$, $\overline{WR}$	t_{CC1}	$7{,}5t_{CY}-200$	480	–	ns
Control pulse duration $\overline{PSEN}$	t_{CC2}	$6t_{CY}-200$	350	–	ns
Data set-up time before $\overline{WR}$	t_{DW}	$6{,}5t_{CY}-200$	390	–	ns
Data set-up time after $\overline{WR}$	t_{WD}	$t_{CY}-50$	40	–	ns
Data hold time					
$\overline{RD}$, $\overline{PSEN}$	t_{DR}	$1{,}5t_{CY}-30$	0	110	ns
$\overline{RD}$ to data input	t_{RD1}	$6t_{CY}-170$	–	350	ns
$\overline{PSEN}$ to data input	t_{RD2}	$4{,}5t_{CY}-170$	–	190	ns
Address set-up time to $\overline{WR}$	t_{AW}	$5t_{CY}-150$	300	–	ns
Address set-up time to data input ($\overline{RD}$)	t_{AD1}	$10{,}5t_{CY}-220$	–	730	ns
Address set-up time to data input ($\overline{PSEN}$)	t_{AD2}	$7{,}5t_{CY}-200$	–	460	ns
Address floating to $\overline{RD}$, $\overline{WR}$	t_{AFC1}	$2t_{CY}-40$	140	–	ns
Address floating to $\overline{PSEN}$ (note 3)	t_{AFC2}	$5t_{CY}-40$	10	–	ns
ALE to control pulse $\overline{RD}$, $\overline{WR}$	t_{LAFC1}	$3t_{CY}-75$	200	–	ns
ALE to control pulse $\overline{PSEN}$	t_{LAFC2}	$1{,}5t_{CY}-75$	60	–	ns
Control pulse to ALE $\overline{RD}$, $\overline{WR}$, $\overline{PROG}$	t_{CA1}	$t_{CY}-40$	50	–	ns
Control pulse to ALE $\overline{PSEN}$	t_{CA2}	$4t_{CY}-40$	320	–	ns
Port control set-up to $\overline{PROG}$	t_{CP}	$1{,}5t_{CY}-80$	50	–	ns
Port control hold to $\overline{PROG}$	t_{PC}	$4t_{CY}-260$	100	–	ns
$\overline{PROG}$ to Port 2 input must be valid	t_{PR}	$8{,}5t_{CY}-120$	–	650	ns
Input data hold time from $\overline{PROG}$	t_{PF}	$1{,}5t_{CY}$	0	150	ns
Output data set-up time	t_{DP}	$6t_{CY}-290$	250	–	ns
Output data hold time	t_{PD}	$1{,}5t_{CY}-90$	40	–	ns
$\overline{PROG}$ pulse duration	t_{PP}	$10{,}5t_{CY}-250$	700	–	ns
Port 2 I/O data set-up time to ALE	t_{PL}	$4t_{CY}-200$	160	–	ns
Port 2 I/O data hold time to ALE	t_{PL}	$1{,}5t_{CY}-120$	15	–	ns

A.C. CHARACTERISTICS (continued)

parameter	symbol	f(t_{CY}) (note 2)	11 MHz		unit
			min.	max.	
Port output from ALE	t_{PV}	$4{,}5t_{CY}+100$	–	510	ns
T0 repetition rate	t_{0PRR}	$3t_{CY}$	270	–	ns
Cycle time	t_{CY}	$1/(f_{XTAL} \times 15)$	1,36	15	μs
MAF80A48H/A35HL; MAF80A49H/A39HL; MAF80A50H/A40HL					
Clock period (note 2)	t_{CY}	$1/(f_{XTAL})$	100	1000	ns

Notes to A.C. characteristics

1. Control outputs: C_L = 80 pF.
   Bus outputs: C_L = 150 pF.
2. f(t_{CY}) assumes 50% duty cycle on XTAL 1 and XTAL 2; minimum frequency = 1 MHz.
3. Bus high-impedance load: 20 pF.



A.C. testing inputs are driven at 2,4 V for a logic 1 and 0,45 V for a logic 0. Output timing measurements are taken 2,0 V for a logic 1 and 0,8 V for a logic 0.

Fig. 13 A.C. testing input, output waveform.

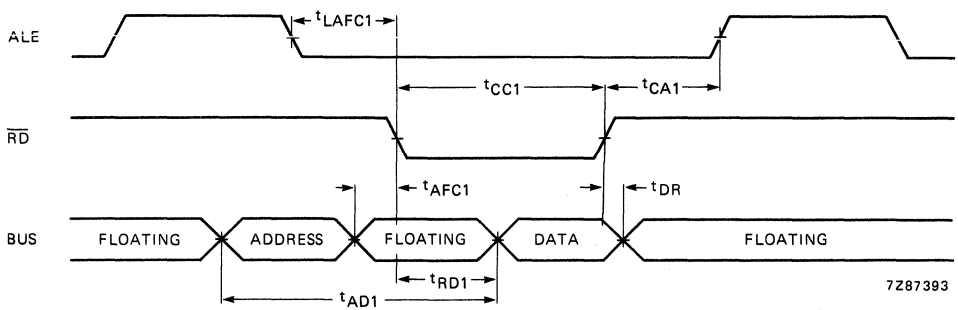


Fig. 14 Read from external data memory.

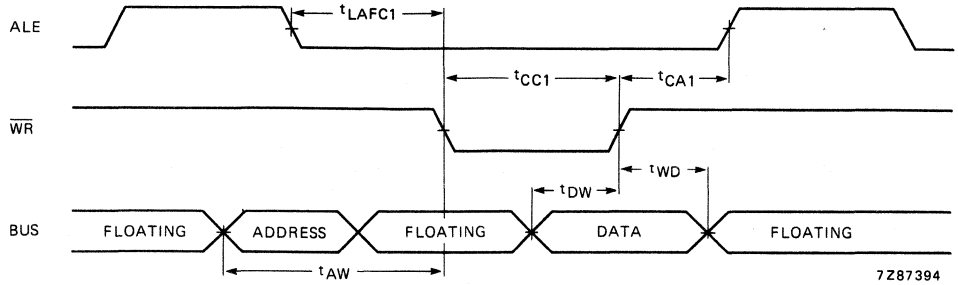


Fig. 15 Write to external memory.

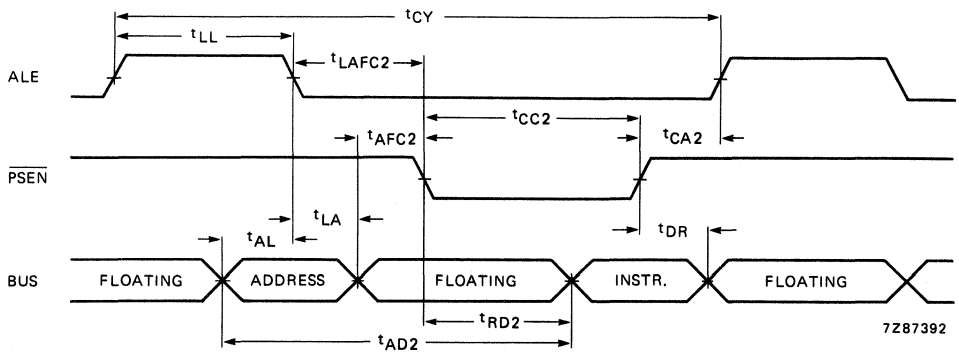


Fig. 16 Instruction fetch from program memory.

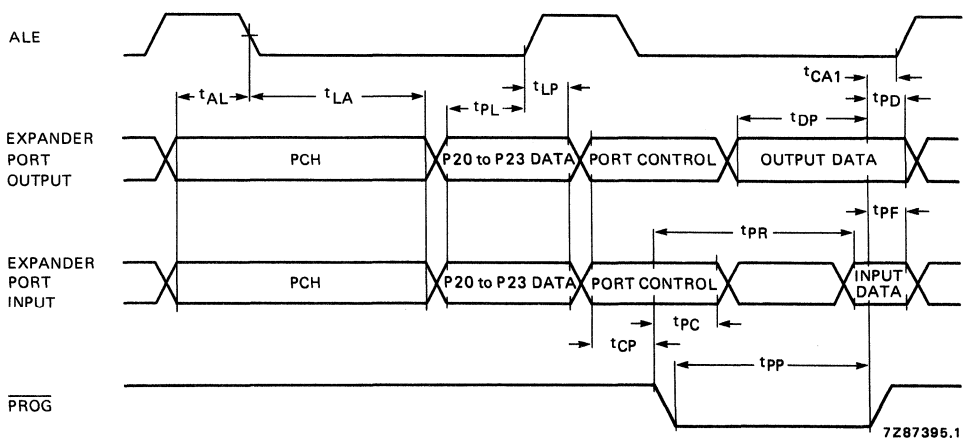


Fig. 17 Port 2 timing.

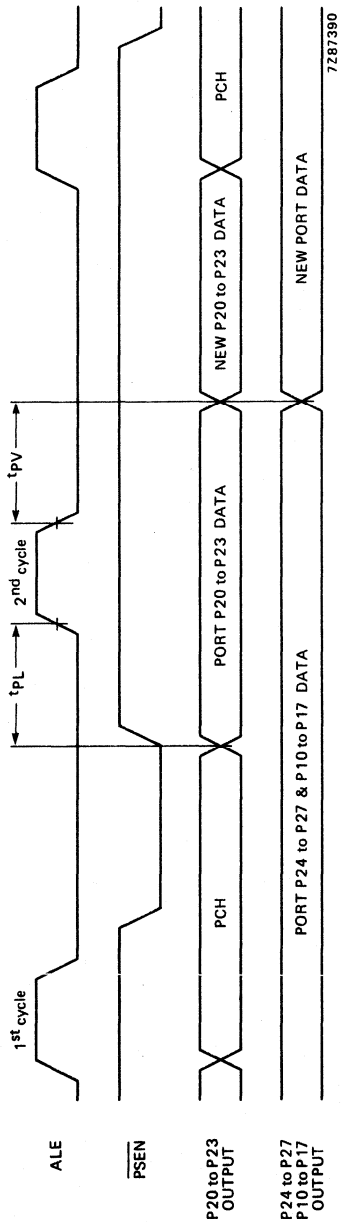


Fig. 18 I/O port timing.

SINGLE-CHIP 8-BIT MICROCONTROLLER

GENERAL DESCRIPTION

The MAB8041A is a universal peripheral interface addition to the MAB8048H family of single-chip 8-bit microcontroller fabricated in NMOS. The device functions as a slave processor and has an asynchronous data register for interface to a MAB8048H master processor; otherwise, it is identical to the MAB8048H. In this slave processor role, the MAB8041A is able to double the computing power and I/O line availability of an MAB8048H system.

The design is based on a low-cost, single-chip microcontroller with program memory, data memory, CPU, I/O, event timer and clock oscillator in a single 40-pin package. The special data bus buffers enables the MAB8041A to function as a peripheral to a 8-bit master processor.

Features

- 8-bit CPU, ROM, RAM, I/O in a single 40-pin package
- 1 K x 8 ROM, 64 x 8 RAM, 18 I/O lines
- 8-bit interval timer/event counter
- Internal oscillator, clock driver
- Single-level interrupts: external and counter/timer
- 17 internal registers: accumulator, 16 addressable registers
- 93 instructions: over 70% single byte
- All instructions 1 or 2 cycles (cycle time 2,5 μs at 6 MHz)
- Easily expandable memory and I/O
- TTL compatible inputs and outputs
- Single 5 V supply
- One 8-bit data bus buffer status register (STS)
- Two 8-bit data bus buffer registers (DBBIN, DBBOUT)
- RAM power-down capability
- DMA interrupt or polled-operation support

QUICK REFERENCE DATA

Supply voltage	$V_{CC} = V_{DD}$	typ.	5 V
Supply current (total)	$I_{CC} + I_{DD}$	max.	125 mA
Cycle time (6 MHz XTAL)	t_{CY}	min.	2,5 μs
Program memory (internal)			1024 x 8-bit
Data memory (internal)			64 x 8-bit

Applications

- Peripheral interfaces and controllers
- Test and measuring instruments
- Sequencers
- Modems and data enciphering
- Environmental control systems
- Audio/video systems

PACKAGE OUTLINES

MAB8041AP: 40-lead DIL; plastic (SOT-129).

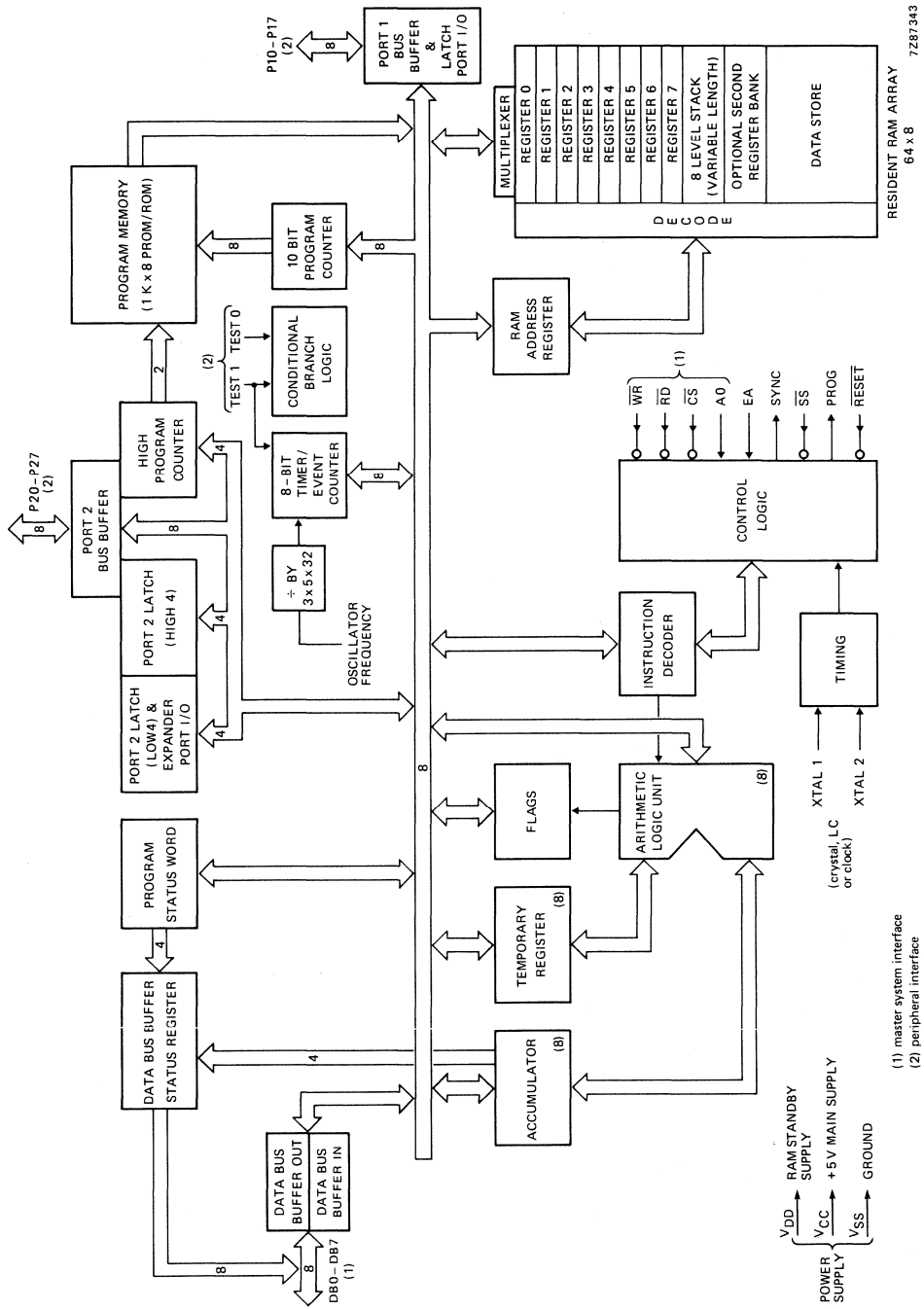


Fig. 1 Block diagram.

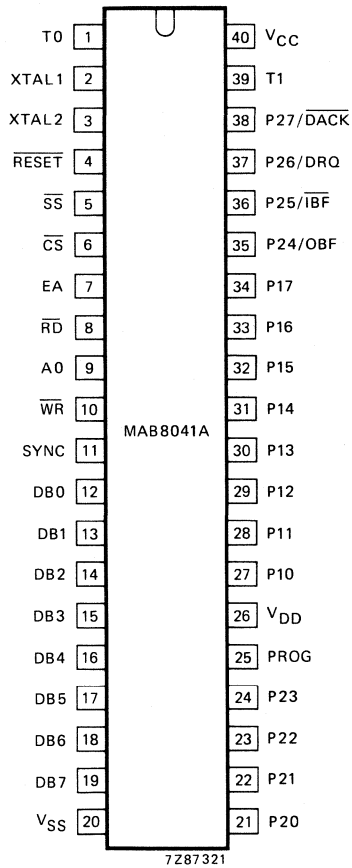


Fig. 2 Pinning diagram; for pin designation see next page.

PINNING

12-19	DB0-DB7	**Data Bus:** three-state, bidirectional DATA BUS BUFFER lines to interface the MAB8041A to an 8-bit master system data bus.
27-34	P10-P17	**Port 1:** 8-bit quasi-bidirectional I/O port (note 1).
21-24	P20-P27	**Port 2:** 8-bit quasi-bidirectional I/O port (note 1).
35-38		P20-P23 interface directly to the 8243 I/O expander device and contain address and data information during port 4-7 access. P24-P27 can be programmed to provide interrupt request and DMA handshake capability. Software control can configure P24 as Output Buffer Full (OBF) interrupt, P25 as Input Buffer Full ($\overline{IBF}$) interrupt, P26 as DMA Request (DRQ), and P27 as DMA ACKnowledge ($\overline{DACK}$).
25	PROG	**Output strobe:** during I/O expander access, it acts as an address/data strobe.
1	T0	**Test 0:** test input pin, directly tested by conditional branch conditions JT0 and JNT0.
39	T1	**Test 1:** test input pin, directly tested by conditional branch conditions JT1 and JNT1. Frequency reference: T1 also functions as an input to the 8-bit event timer, under software control.
6	$\overline{CS}$	**Chip select input:** selects one MAB8041A microcontroller from several connected to a common data bus.
4	$\overline{RESET}$	**Reset input:** resets status flip-flops and sets the program counter to zero. During program verification the address is latched by a LOW to HIGH transition on $\overline{RESET}$ and data at the address is output on the data bus (note 2).
11	SYNC	**Clock output:** output signal occurs once per MAB8041A instruction cycle. It can be used to synchronize external circuitry, to synchronize single step operation or as a general-purpose clock output.
8	$\overline{RD}$	**Read input:** enables the master CPU to read data and status words from the DATA BUS BUFFER OUT or DATA BUS BUFFER STATUS REGISTER (active LOW).
10	$\overline{WR}$	**Write input:** enables the master CPU to write data and command words to the DATA BUS BUFFER IN (active LOW).
7	EA	**External access input:** allows emulation, testing and PROM/ROM verification. When HIGH, forces instruction fetch from external memory.
9	A0	**Command/Data select input:** address input used by the master processor to indicate whether byte transfer is data (A0 = 0) or command (A0 = 1).
2 3	XTAL 1 XTAL 2	**Crystal inputs:** inputs for a crystal, LC-network or an external timing signal to determine the internal oscillator frequency (note 2).
5	$\overline{SS}$	**Single step input:** not used.

40	V_{CC}	**Power supply:** + 5 V main power supply pin.
26	V_{DD}	**Power supply:** + 5 V during normal operation; low power standby pin.
20	V_{SS}	**Ground:** circuit earth potential.

Notes

1. Each port line can be designated as an input or an output. A line is designated as an input by first writing a logic 1 to the line. $\overline{\text{RESET}}$ sets all lines to logic 1.
2. Non-standard TTL V_{IH}.

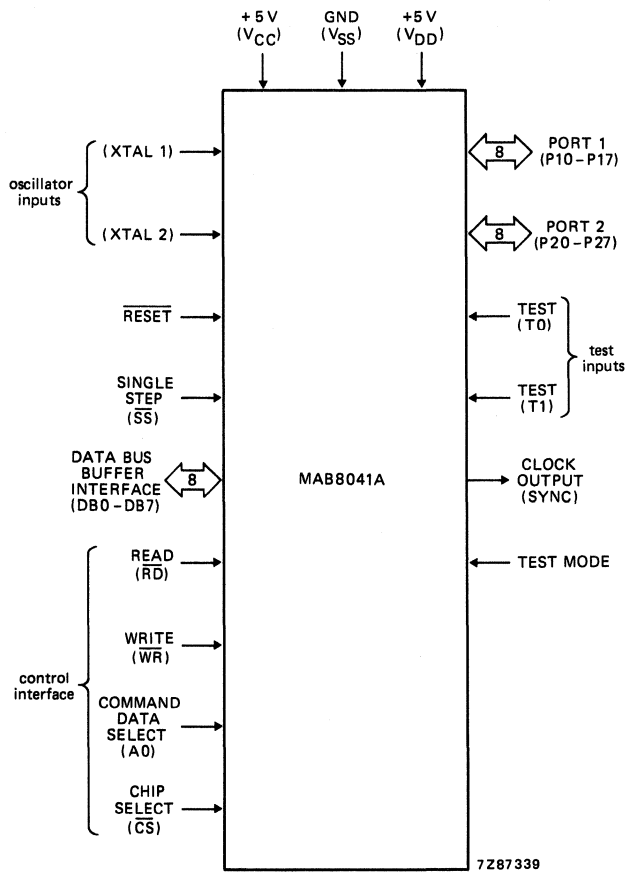


Fig. 3 Functional diagram.

FUNCTIONAL DESCRIPTION

The following sections provide a detailed functional description of the MAB8041A microcontroller as shown in the block diagram, Fig. 1.

Central processing unit

The central processing unit (CPU) performs basic data operations and controls data flow in the microcontroller via the internal 8-bit data bus. The CPU comprises the following functional blocks:

- Arithmetic logic unit (ALU)
- Instruction decoder
- Accumulator
- Flags

Arithmetic logic unit (ALU)

The ALU capabilities are as follows:

- ADD with or without carry
- AND, OR and EXCLUSIVE OR
- Increment, Decrement
- Bit complement
- Rotate left or right
- Swap
- BCD decimal adjust

Typically data from the accumulator is combined in the ALU with data from some other source on the microcontroller internal bus (from a register or an I/O port). The result of an ALU operation can be transferred to the internal bus or returned to the accumulator.
The CARRY flag is used as an indicator when an operation such as ADD or ROTATE requires more than 8 bits. Similarly, during decimal adjustment and other BCD operations the AUXILIARY CARRY flag can be set and acted upon. These flags are part of the Program Status Word (PSW).

Instruction decoder

The instruction decoder stores and decodes the operation code (op-code) part of the program instruction during an instruction fetch. The decoder generates outputs which are used in combination with various timing signals to control all functions performed in the ALU. The instruction decoder also controls the source and destination of ALU data.

Accumulator

The accumulator is the primary source of data to the ALU and is also the destination for most results. Data between I/O ports and memory normally passes through the accumulator.

Program memory (see Fig. 4)

The resident program memory consists of 1024 8-bit words in a read-only memory (ROM). Each location is directly addressable by the 10-bit program counter. The program memory is divided into location 'pages', each of 256 bytes. Three program memory locations are reserved for special use. These contain the first instruction to be executed after one of three events. Interrupt vectors cause instructions to be fetched from these locations as follows:

- Location 0; after a $\overline{\text{RESET}}$ input to the processor
- Location 3; after an interrupt by an Input Buffer Full (IBF) condition (when IBF is enabled)
- Location 7; when a timer/event counter overflow interrupt is enabled

FUNCTIONAL DESCRIPTION (continued)

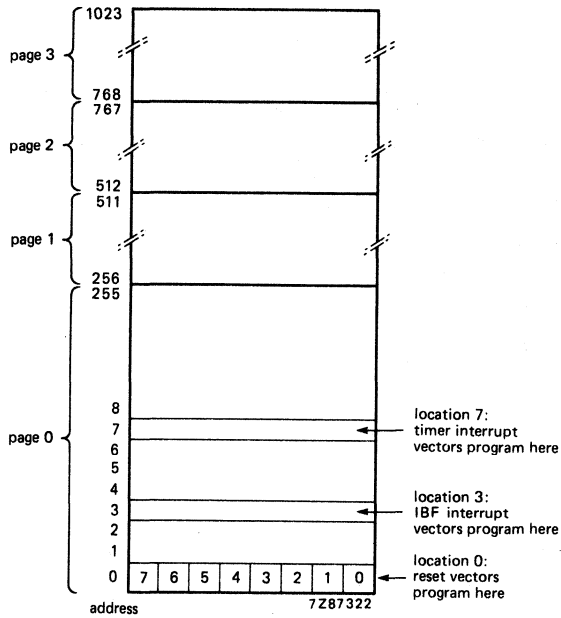


Fig. 4 Program memory map.

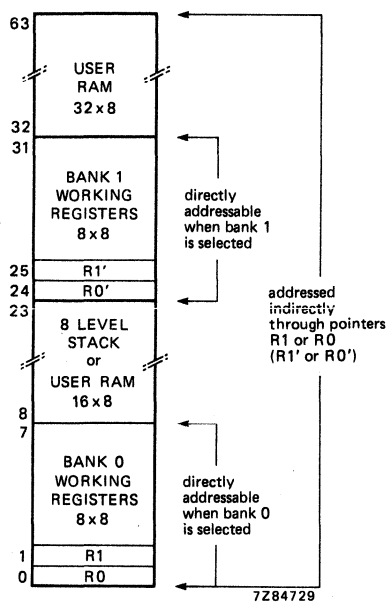


Fig. 5 Data memory map.

After a system $\overline{\text{RESET}}$, program execution starts at location 0. Instructions in program memory are normally executed sequentially. Program control can be transferred out of the main line of code as follows:

- By an input buffer full (IBF) interrupt
- By a timer/event counter overflow interrupt
- When a jump or call instruction is present

An IBF interrupt (if enabled) automatically transfers control to location 3 while a timer interrupt transfers control to location 7.

All conditional JUMP instructions and the indirect JUMP instruction are limited in range to the current 256-location page (only PC bits 0 to 7 changed). A conditional or indirect JUMP starting in location 255 of a page must reference a destination on the following page. Program memory can be used to store constants as well as program instructions. Instruction MOVP3 is specifically designed for efficient transfer of look-up table information from page 3 of memory.

Data memory (see Fig. 5)

The resident data memory consists of a 64-byte random-access data memory (RAM). It contains two working register banks, an 8-level program counter stack and a user scratch-pad memory. The amount of scratch-pad memory depends upon the number of addresses nested in the stack and the number of working registers in use.

Addressing data memory

Working registers R0 to R7 occupy the first eight locations in the RAM (bank 0). They can be addressed directly by specifying a register number in the instruction. Ease of addressing makes these locations suitable for storing frequently accessed intermediate results. Other locations in data memory are addressed indirectly by using R0 or R1 to specify the required address. As all RAM addresses are confined to 6 bits, the two most significant bits (6 and 7) of the addressing registers are ignored.

Working registers

There are two banks of eight working registers in the data memory:

- Bank 0; locations 0 to 7
- Bank 1; locations 24 to 31

Bank 0 is automatically selected by a RESET signal, whereupon references to R0 to R7 in MAB8041A instructions operate on locations 0 to 7 in data memory. A 'select register bank' instruction is used to chose between banks during program execution. When SEL RB1 (Select Register Bank 1) is executed program references to R0 to R7 operate on locations 24 to 31. Registers 0 and 1 in the active register bank are used as indirect address registers for all locations in data memory.

Register bank 1 is usually reserved for handling interrupt service routines, thus preserving the contents of the main program registers. The SEL RB1 instruction can be sent at the start of an interrupt service routine. On return to the main program, an RETR (return and restore status) instruction will then automatically restore the previously selected bank. During interrupt processing, registers in bank 0 can be accessed indirectly using R0' and R1'.

When register bank 1 is not in use, locations 24 to 31 serve as additional scratch-pad memory.

FUNCTIONAL DESCRIPTION (continued)

Program counter stack

Locations 8 to 23 are designated as the program counter stack, which allows up to eight levels of subroutine nesting. A subroutine may call a second subroutine, which in turn may call a third, up to the eighth level. Any unused stack location can be used as scratch-pad memory, each level providing two additional RAM locations.

When program control passes temporarily from the main program to a subroutine or interrupt service routine, the 10-bit program counter and bits 4 to 7 of the program status word (PSW) are stored in two stack locations. When a RETR instruction returns control to the main program, the program counter and PSW bits 4 to 7 are restored. However returning via an RET (return) instruction does *not* restore the PSW bits. The program counter stack is addressed by three stack pointer bits (bits 0 to 2) in the PSW.

10-bit program counter

The program counter (PC) is a 10-bit counter/register that points to the location from which the next instruction is to be fetched. The program counter can address any of the 1024 locations in program memory and is normally incremented sequentially for each instruction. When control is temporarily passed from the main program to a subroutine or interrupt routine, the PC contents must be altered to point to the address of the desired routine. The program counter stack (Fig. 6) is used to save the current PC contents so that, at the end of the routine, main program execution can continue. A $\overline{\text{RESET}}$ signal initializes the program counter to zero.

Program counter stack (see Fig. 6)

The program counter stack comprises 16 locations (8 to 23) in data memory, used to store the 10-bit program counter and 4 bits of the program status word (PSW). An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 pairs of the program counter stack. The pair to be used is determined by a 3-bit pointer; part of the PSW. The stack pointer is initialized by a $\overline{\text{RESET}}$ signal to 00H, which corresponds to RAM locations 8 and 9.
The first CALL or interrupt results in the program counter contents being transferred to locations 8 and 9 of the RAM. The stack pointer is automatically incremented by 1 point to locations 10 and 11 of another CALL.

Nesting of subroutines within subroutines can continue up to 8 times without overflowing the stack. If overflow does occur the deepest address stored (locations 8 and 9) will be overwritten and lost since the stack pointer overflows from 07H to 00H. It also underflows from 00H to 07H.

The end of a subroutine, which is signalled by a return instruction RET or RETR, causes the stack pointer to be decremented and the contents of the resulting pair to be transferred to the program counter.

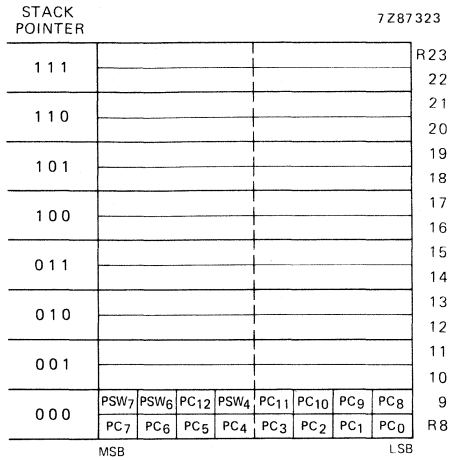


Fig. 6 Program counter stack.

Program status word (see Fig. 7)

The 8-bit program status word (PSW) stores information about the current status of the microcontroller. The PSW bits are:

- Bits 0 to 2 — stack pointer bits (SP_0, SP_1, SP_2)
- Bit 3 — not used
- Bit 4 — working register bank select (RBS); 0 = register bank 0; 1 = register bank 1
- Bit 5 — flag 0 (F0); general purpose flag cleared or complemented and tested with conditional jump instructions. It may be used during data transfer to an external processor
- Bit 6 — auxiliary carry (AC); half-carry bit generated by an ADD instruction and used by the decimal adjust instruction DA A
- Bit 7 — carry (CY); the carry flag indicates that previous operation has resulted in an overflow of the accumulator

The PSW is a collection of flip-flops located throughout the machine, which are read or written as a whole. The PSW can be loaded to or from the accumulator by the MOV A, PSW or MOV PSW, A instructions. The ability to write directly to the PSW allows easy restoration of machine status after a power-down sequence. Bits 4, 5, 6, and 7 are stored in the program counter stack with every subroutine and interrupt vector. These bits are restored in the PSW with a RETR (return and restore) instruction which must be used at the end of an interrupt and can be used at the end of a normal subroutine. The RET instruction has no restore feature and cannot be used at the end of an interrupt.

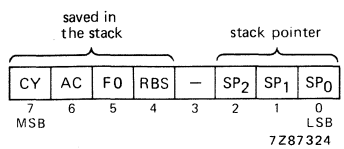


Fig. 7 Program status word.

FUNCTIONAL DESCRIPTION (continued)

Conditional branch logic

Conditional branch logic in the MAB8041A allows the status of various processor flags, inputs and other hardware functions to affect directly program execution. Sampling of the status is performed in state 3 of the first cycle.
Table 1 lists internal test conditions and conditional jump instructions used to change the program execution sequence. The destination address must be within the page of program memory (256 locations) in which the jump instruction occurs.

Table 1 Conditional branches

test	jump condition	jump instruction
accumulator	all bits zero	JZ
	any bit non-zero	JNZ
accumulator bit test	bit b = 1	JBb
carry flag	1	JC
	0	JNC
user flag F0	1	JF0
user flag F1	1	JF1
timer overflow flag	1	JTF
test input T0	1	JT0
	0	JNT0
test input T1	1	JT1
	0	JNT1
input buffer flag	0	JNIBF
output buffer flag	1	JOBF

Oscillator and timing circuits

The MAB8041A contains its own internal oscillator and clock driver. A crystal, LC network or external pulse generator determines the oscillator frequency.

Oscillator (see Fig. 8)

The internal oscillator is a series resonant circuit with a frequency range of 1 to 6 MHz. Pin 2 (XTAL 1) and pin 3 (XTAL 2) are input and output respectively of a high gain amplifier stage. A crystal or LC-network connected across these pins provides feedback and phase shift for oscillation. The recommended connections are shown in Fig. 9 for a crystal oscillator, in Fig. 10 for the LC oscillator mode and in Fig. 11 for driving from an external source.

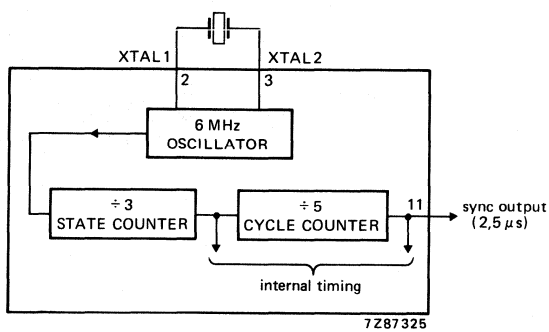


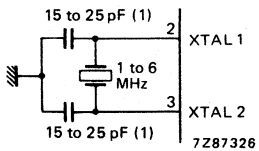
Fig. 8 Oscillator block diagram.

State counter (see Fig. 8)

The 6 MHz oscillator output is divided by three in the state counter to define the state times of the machine. Each instruction cycle consists of 5 states as shown in Fig. 12 and Table 2. The overlap of address and execution operations allows fast instruction execution.

Cycle counter (see Fig. 8)

The output of the state counter is divided by five in the cycle counter to group the five states into instruction cycles of 2,5 μs duration. The sync output at pin 11 is available for synchronizing external circuits or as a general-purpose clock output.



(1) Including crystal-socket stray capacitance.

Fig. 9 Crystal oscillator mode. Crystal series impedance should be $< 75\ \Omega$ at 6 MHz and $< 180\ \Omega$ at 3,6 MHz.

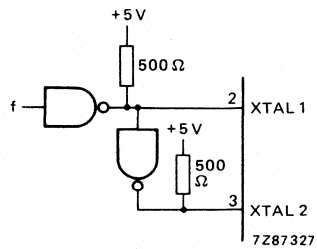
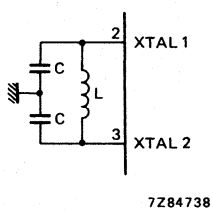


Fig. 11 Driving from external source. Both XTAL 1 and XTAL 2 should be driven. Resistors to V_{CC} (+ 5 V) are needed to ensure $V_{IH} = 3{,}8$ V if TTL circuitry is used.



L (μH)	C (pF)	nominal frequency (MHz)
45	20	5
130	20	3

Fig. 10 LC oscillator mode. Each C should be ≈ 20 pF including stray capacitance. $C_{pp} \approx 5$ to 10 pF (pin-to-pin capacitance).

$$C' = \frac{C + 3C_{pp}}{2} \qquad f \approx \frac{1}{2\pi\sqrt{LC'}}$$

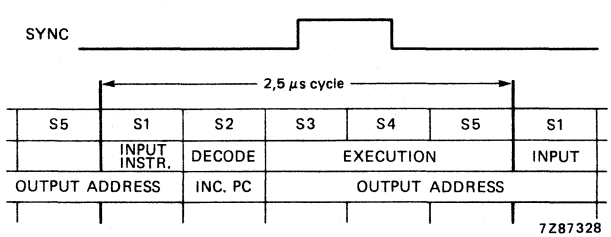


Fig. 12 Instruction cycle timing.

Table 2 Instruction timing (see also Fig. 12)

	cycle 1					cycle 2				
instruction	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,P	fetch instruction	increment program counter	–	increment timer	–	–	read port	–	–	–
OUTL P_p,A	| |	| |	–	| |	output to port	–	–	–	–	–
ANL P_p,data	| | |	| | |	–	| | |	read port	fetch immediate data	–	increment program counter	output to port	–
ORL P_p,data	| | |	| | |	–	| | |	read port	fetch immediate data	–	increment program counter	output to port	–
IN A,DBB	| |	| |	–	| |	–					
OUT DBB,A	| |	| |	–	| |	output to port					
MOVD A,P	fetch instruction	increment program counter	output opcode/ address	increment timer	–	–	read P2 lower	–	–	–

Continued on next page.

Table 2 Instruction timing (continued)

	cycle 1					cycle 2				
instruction	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
MOVD P_p,A	fetch instruction	increment program counter	output opcode/ address	increment timer	output data to P2 lower	–	–	–	–	–
ANLD P_p,A	|	|	output opcode/ address	|	output data	–	–	–	–	–
ORLD P_p,A	|	|	output opcode/ address	|	output data	–	–	–	–	–
J (conditional)	|	|	sample condition	increment timer	–	fetch immediate data		update program counter	–	–
STRT CNT STRT T	|	|	–	–	start counter					
STOP TCNT	|	|	–	–	stop counter					
EN I	|	|	–	enable interrupt	–					
DIS I	|	|	–	disable interrupt	–					
EN DMA	|	|	–	DMA enabled DRQ cleared	–					
EN FLAGS	fetch instruction	increment program counter	–	OBF, $\overline{\text{IBF}}$ output enabled	–					

FUNCTIONAL DESCRIPTION (continued)

8-bit timer/event counter (see Fig. 13)

The internal 8-bit timer/event counter has several software selectable modes of operation. As an interval timer it can generate accurate delays from 80 μs to 20,48 ms without unduly loading the processor. In the event counter mode, external events such as switch closures or tachometer pulses can be counted and used to direct program flow.

Timer configuration

An 8-bit register counts pulses from either the internal clock and prescaler or from an external input. The counter can be preset and read with two MOV instructions; MOV T,A transfers the contents of the accumulator to the counter, and MOV A,T transfers the contents of the counter to the accumulator. The counter is initialized by the MOV T,A instruction and is not cleared by a $\overline{\text{RESET}}$ signal. The counter is stopped by a $\overline{\text{RESET}}$ or STOP TCNT instruction and remains so until restarted either as a timer (STRT T) or as a counter (STRT CNT). When started the counter will increment to its maximum count (FFH), then overflow to zero and continue to count until stopped by STOP TCNT or $\overline{\text{RESET}}$.
The increment from maximum count to zero (overflow) sets the timer flag TF and generates an interrupt request. The state of the overflow flag can be tested with the conditional jump instruction JTF. The flag is reset by executing a JTF instruction or by a $\overline{\text{RESET}}$ signal.
The timer interrupt request is stored in a latch and ORed with the input buffer full interrupt request (IBF). The timer interrupt can be enabled or disabled independent of the IBF interrupt by the EN TCNTI and DIS TCTNI instructions. Enabling the counter overflow causes a subroutine call to location 7 which contains the timer service routine. If the timer and IBF interrupt occur simultaneously, the IBF source is recognized and the call will be to location 3. The timer interrupt is latched and held until the data bus buffer input register has been serviced. On return from the service routine, the timer interrupt is immediately recognized. A pending timer interrupt is reset by the initiation of a timer interrupt service routine.

Event counter mode

The STRT CNT instruction connects the TEST 1 input pin (T1) to the counter input to enable the counter. This instruction does not clear the counter. When using a 6 MHz crystal, HIGH-to-LOW transitions of T1 will increment the counter at a maximum rate of one per three instruction cycles (7,5 μs). No minimum frequency limit is imposed, but T1 must remain HIGH for a minimum of 500 ns during a count cycle.

Timer mode

The STRT T instruction connects an internal clock to the counter input to enable the counter. An input frequency of 12,5 kHz is derived from a divide-by-32 prescaler and driven by the 400 kHz machine cycle clock. The 12,5 kHz input increments the counter every 80 μs. This allows delays and timing sequences between 80 μs and 20,48 ms to be generated with a minimum of software timing loops. Time intervals longer than 20 ms can be measured by accumulating multiple overflows in a register under software control. For time intervals less than 80 μs, an external clock can be applied to the TEST 1 input using the event counter mode of operation. The 2,5 μs SYNC output (see Fig. 8) divided by 3 or more can serve as the external clock, software loops being used to compensate long delays generated by the timer.

TEST 1 (T1) event counter input

The input TEST 1 is multifunctional. As a test input it is automatically initialized by a $\overline{\text{RESET}}$ signal and can be tested using the processor conditional branch instructions.
In the event counter mode, TEST 1 is internally switched under control of the STRT CNT instruction, via an edge connector, to the 8-bit counter. This instruction does not prevent TEST 1 being tested via conditional jump instructions. In this mode, TEST 1 is sampled once per instruction cycle. After a HIGH level is detected, the next LOW level will increment the counter by one. STOP TCNT stops this function and restores TEST 1 as a test input.

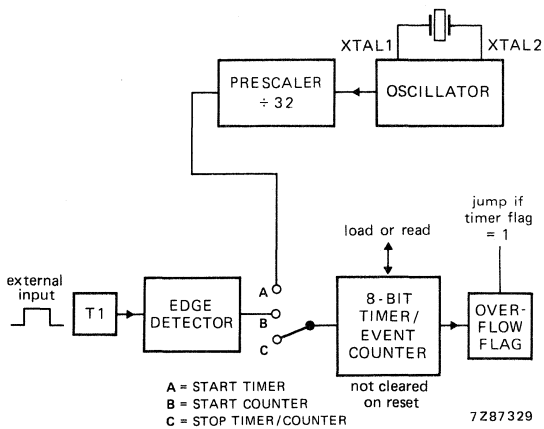


Fig. 13 8-bit timer/event counter.

Test inputs

TEST 0 (T0) and TEST 1 (T1) are multifunctional test inputs. Table 3 lists the conditional jump instructions to directly test the status of these inputs during normal operation.

Table 3 Test inputs status test

test	jump condition	jump instruction
test input T0	1	JT0
test input T0	0	JNT0
test input T1	1	JT1
test input T1	0	JNT1

An external logic signal applied to test inputs T0 or T1 is sampled on execution of the appropriate conditional jump instruction. The path of program execution is dependent on the state of the external logic signal when sampled. Both test inputs are TTL compatible.

FUNCTIONAL DESCRIPTION (continued)

Interrupts (see Fig. 14)

The MAB8041A has two internal interrupts:

- Input buffer full (IBF)
- Timer overflow

IBF forces a CALL to location 3 in program memory. It is enabled by EN 1 and disabled by DIS 1.
Timer overflow forces a CALL to location 7 in program memory. It is enabled by EN TNCTI and disabled by DIS TCNTI.
An IBF interrupt request is generated when $\overline{WR}$ and $\overline{CS}$ are LOW, regardless of whether interrupts are enabled. The interrupt request is cleared only on entering the IBF service routine. Thus a DIS I instruction does not clear a pending IBF interrupt.

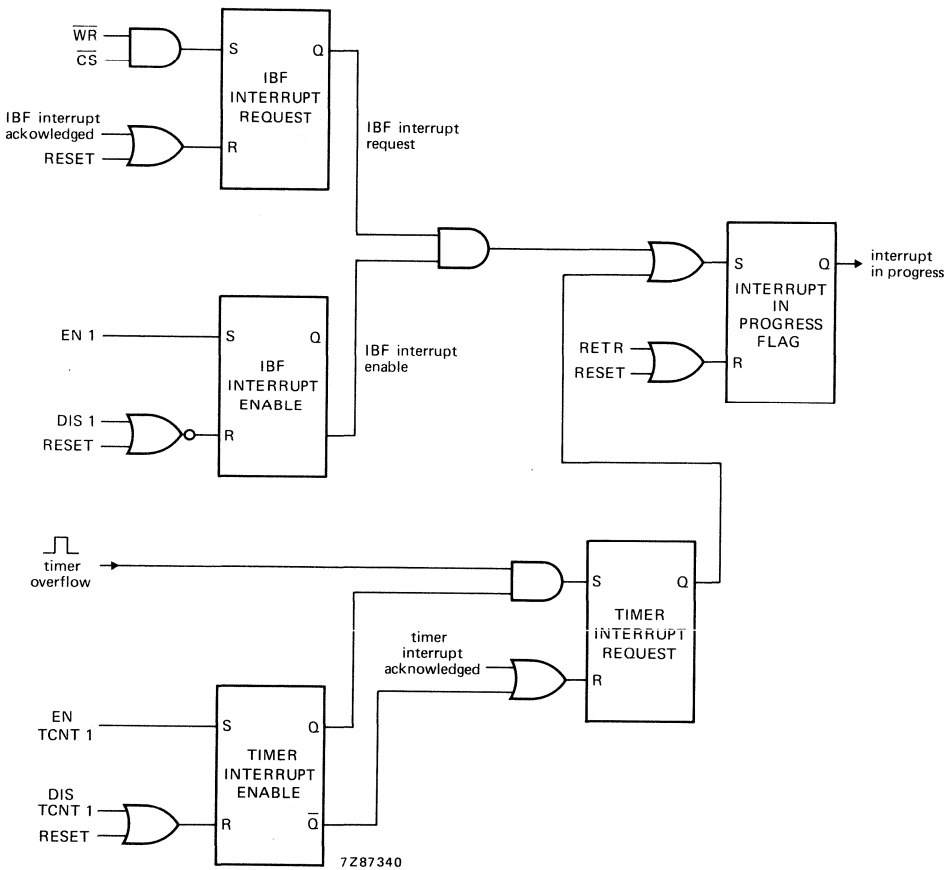


Fig. 14 Internal interrupt logic.

Interrupt timing latency

The interrupt service latency time is the sum of the existing instruction time, the interrupt acknowledge time and the internal call to the interrupt vector address. For the MAB8041A it is from 4 to 7 clock cycles. When the IBF interrupt is enabled and an interrupt request occurs, the following interrupt sequence starts immediately the existing executing instruction ends.

- A CALL to location 3 is forced
- The program counter and bits 4 to 7 of PSW are stored in the stack
- The stack pointer is incremented

Location 3 in program memory should contain an unconditional jump to the beginning of the IBF interrupt service routine stored elsewhere in program memory. A return and restore status (RETR) instruction returns control to the main program at the end of the service routine. RETR restores the program counter and bits 4 to 7 of the PSW, plus automatic restoration of the previously active register bank. Interrupts are also re-enabled by RETR. When a timer overflow interrupt is enabled by EN TCNT I, the interrupt routine is started by a timer/counter register overflow. A CALL to location 7 in program memory is forced and the sequence proceeds as described above.

Interrupt timing

Interrupt inputs, under program control, may be enabled by EN I and EN TCNT I instructions and disabled by DIS I and DIS TCNT I instructions. A $\overline{\text{RESET}}$ input will also disable interrupts. An interrupt request must be removed before the RETR instruction is executed to return from the service routine, otherwise the processor will immediately re-enter the service routine.
Since the interrupt system is single level, once an interrupt is detected, all further interrupt requests are latched pending a RETR instruction to re-enable the interrupt input logic. This occurs at the start of the second cycle of the RETR instruction. If an IBF interrupt and a timer overflow interrupt occur simultaneously, the IBF is acknowledged first and the timer overflow remains pending until the end of the interrupt service routine.

External interrupts

An external interrupt can be generated by using the timer/counter in the event counter mode. The counter is first preset to (FFH), then EN TCNT I instruction is executed. A timer overflow interrupt is generated by the first HIGH-to-LOW transition of the TEST 1 input. If an IBF interrupt occurs during servicing of the timer/counter interrupt, the IBF interrupt remains pending until the service routine is completed.

Master interrupts and DMA

Two external interrupts are available to the master system using the EN FLAGS instruction, which allocates two I/O lines (P24, P25) on PORT 2. P24 is the output buffer full (OBF) interrupt request line to the master system, indicating the internal status of the OBF flag. P25 is the input buffer empty interrupt request line indicating the internal status of the IBF inverted flag. Writing a logic 0 to these port pins inhibits the outputs and a logic 1 will enable them. The interrupts are typically enabled after 'power on' as the I/O ports are set at logic 1. The effect of the EN FLAG is only cancelled by a device RESET.
Direct memory access (DMA) handshake controls are available from two pins (P26, P27) on PORT 2, enabled by the EN DMA instruction. P26 becomes DMA request (DRQ) and P27 becomes DMA acknowledge ($\overline{\text{DACK}}$). DRQ is initiated by writing a logic 1 to P26, the DMA controller transfers the data into DBBIN register using DACK as a chip select. EN DMA is only cancelled by a chip RESET.

FUNCTIONAL DESCRIPTION (continued)

Reset (see Fig. 15)

The RESET input enables internal initialization of the processor. At power-on, an automatic initialization pulse can be generated by connecting a 1 μF capacitor between the $\overline{\text{RESET}}$ input and ground. An internal pull-up transistor charges the capacitor and a Schmitt-trigger circuit generates a clean transition. If an external $\overline{\text{RESET}}$ pulse is used, it must hold the input LOW for a minimum of 10 ms after the power supply is within tolerance.

The functions of the RESET input are as follows:

- Sets the program counter to zero
- Sets the stack pointer to zero
- Selects register bank 0
- Sets PORTS 1 and 2 to the input mode
- Disables interrupts
- Stops the timer
- Clears the timer flag
- Clears status flip-flops F0 and F1

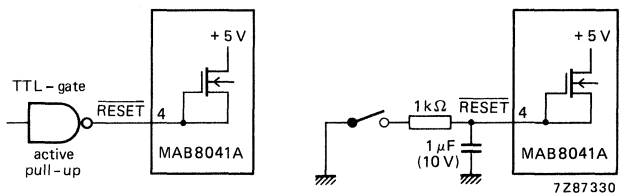


Fig. 15 An external reset is shown in (a) and power-on reset in (b).

Data bus buffer (see Fig. 16)

Two 8-bit data bus buffer registers, DBBIN and DBBOUT, serve as temporary stores for commands and data flowing between the MAB8041A and the master processor. Externally data is transmitted and received by the DBB registers on execution of input or output instructions from the master processor via four control signals:

- A0 Address input signifying control or data signals
- $\overline{\text{CS}}$ Chip Select
- $\overline{\text{RD}}$ Read strobe
- $\overline{\text{WR}}$ Write strobe

Transfer can take place with or without MAB8041A program interference by internal interrupt enable/disable signals.
Internal data transfer between the DBB and the accumulator is software controlled and completely asynchronous to the external processor timing. This allows peripheral control tasks to be handled by software independent of the main processor, while still maintaining a data interface with the master system.

Data bus buffer configuration

The two 8-bit buffer registers, DBBIN and DBBOUT are used for storing data, accessed by the external processor using the $\overline{WR}$ and $\overline{RD}$ lines respectively. The bidirectional three-state data bus can be directly connected to an 8-bit microprocessor system. Data is transferred to and from the DBBIN and DBBOUT registers by the four control lines $\overline{WR}$, $\overline{RD}$, $\overline{CS}$ and A0.
An 8-bit register containing eight status flags is used to indicate the status of the DBB registers.

The eight status flags are defined as follows:

- OBF — The output buffer full flag is automatically set when the MAB8041A loads the DBBOUT register and is cleared when the master processor reads the register
- IBF — The input buffer full flag is set when the master processor writes a character to the DBBIN register and is cleared when the MAB8041A transfers the data register contents to its accumulator
- F0 — This general-purpose flag can be cleared or toggled under MAB8041A software control, and is used to transfer MAB8041A status information to the master processor
- F1 — The command/data flag is set to the condition of the A0 input line when the master processor writes a character to the data register. This flag can also be cleared or toggled under MAB8041A program control.
- ST4 to ST7 — These bits are user-defined status bits (see Table 6 instruction MOV STS, A)

All flags in the status register are cleared by a $\overline{RESET}$ input.

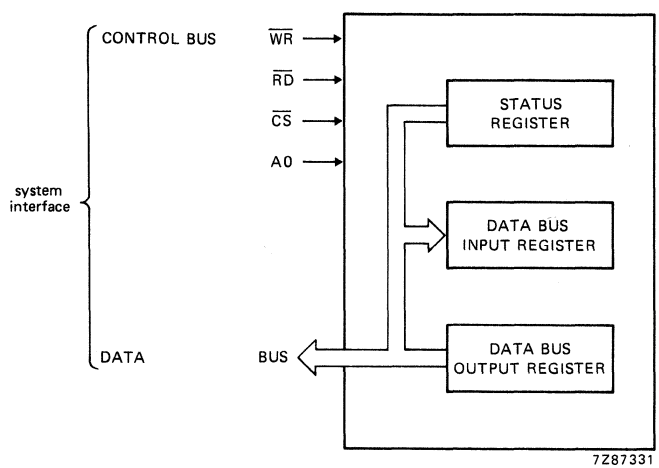


Fig. 16 Data Bus Buffer (DBB) registers.

FUNCTIONAL DESCRIPTION (continued)

System interface (see Fig. 17)

An 8041A can be connected to a standard bus system. Data lines DB0 to DB7 form a three-state bidirectional port for direct connection to the system data bus. The bus interface has sufficient drive capability (400 μs) for small systems without the need for buffers.

Four control signals are required to handle the data and status information transfer:

- $\overline{WR}$ I/O WRITE transfers data from the system bus to the DBBIN register and sets the F1 flag in the status register
- $\overline{RD}$ I/O READ transfers data from the DBBOUT register or status register to the system data bus
- $\overline{CS}$ CHIP SELECT enables one MAB8041A out of several connected to a common bus
- A0 Address input selects either the 8-bit status register or DBBOUT register during an I/O READ. It also sets the F1 flag in the status register during an I/O WRITE.

The $\overline{WR}$ and $\overline{RD}$ signals are active LOW, standard peripheral control signals used to synchronize data transfer between the system bus and peripherals.
The $\overline{CS}$ and A0 signals are decoded from the address bus of the master system. In a system having a limited number of I/O devices a linear addressing configuration can be used, where A0 and A1 lines can be directly connected to A0 and $\overline{CS}$ inputs as shown in Fig. 17.

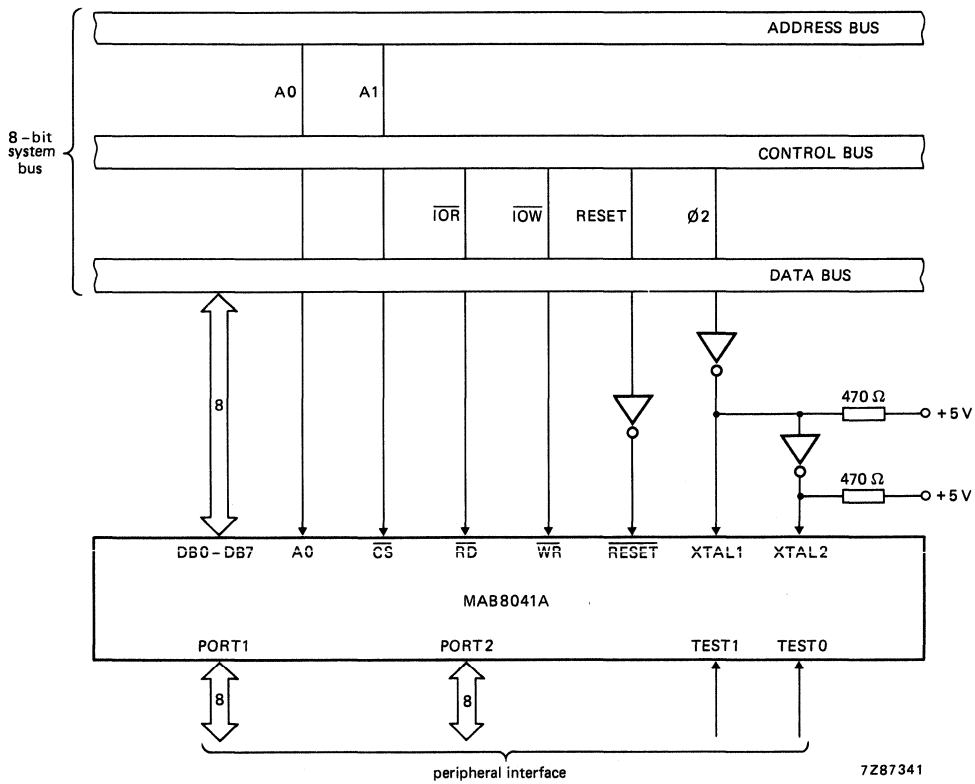


Fig. 17 MAB8041A connected to a standard bus system.

Data read

The relative timing of a DBBOUT read is shown in Table 4. When the lines $\overline{CS}$, A0 and $\overline{RD}$ are LOW, the DBBOUT register contents are placed on the three-state data lines DB0 to DB7 and the OBF flag is cleared.
The master processor uses the four lines $\overline{CS}$, A0, $\overline{WR}$ and $\overline{RD}$ to control data transfer between the DBBOUT register and the master system by the following operations:

Table 4 Data transfer controlled by the master processor

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A0	operations
0	0	1	0	read DBBOUT register
0	0	1	1	read status register
0	1	0	0	write DBBIN data register
0	1	0	1	write DBBIN command register
1	X	X	X	disable DBB

Where X is the don't care state.

Status read

The logic sequence required for a STATUS register read is shown in Table 4. When $\overline{CS}$ and $\overline{RD}$ are LOW and A0 is HIGH the status register contents appear on data lines DB0 to DB7.

Data write

The sequence required for writing information to the DBBIN register is also shown in Table 4. When $\overline{CS}$ and $\overline{WR}$ are LOW the contents of the system data bus are latched into the DBBIN register. The IBF flag is also set and an interrupt generated, if enabled.

Command write

During a write operation the state of the A0 input is latched into the status register in the F1 (command/data) flag location. This extra bit signals whether the DBBIN contents are command (A0 = 1) or data (A0 = 0) information.

Input/output interface

The MAB8041A has 16 lines for input and output functions, grouped as 8-bit TTL-compatible ports (PORT 1 and PORT 2). Each port line can function as either input or output under software control. The I/O capacity can be increased by using the lower four lines of PORT 2 to interface to an 8243 I/O expander device. This feature saves program space and design time, and also improves the bit-handling capability of the MAB8041A.

Ports 1 and 2

Ports 1 and 2 are each 8 bits wide and have identical I/O characteristics. Data that is written to these ports by an OUTL P_p,A instruction is latched and remains unchanged until it is rewritten. When the IN A,P_p instruction is executed, input data is sampled and is therefore present at the PORT until read by an input instruction. Both PORT inputs are fully TTL-compatible and the outputs are capable of driving one standard load.

FUNCTIONAL DESCRIPTION (continued)

Circuit configuration (see Figures 18 and 19)

Fig. 18 shows the special output structure which allows each of the PORT lines to serve as input or output, or both, even though outputs are statically latched.
Each line has a permanent high impedance pull-up of 50 kΩ, which provides sufficient source current for a TTL HIGH level, yet can be pulled LOW by a standard TTL gate drive. When a logic 1 is written to a line, a low impedance pull-up of 5 kΩ is switched in for 500 ns to provide a fast transition from logic 0 to logic 1. When a logic 0 is written to the line, a low impedance pull-down of 300 Ω is active to provide TTL current-sinking capability. A logic 1 must first be written to a PORT pin for it to be used as an input.

Note

A $\overline{\text{RESET}}$ initializes all PORT pins to the high impedance logic 1 state.

An external TTL device connected to a pin has sufficient current-sinking capability to pull the pin down to a LOW state. An IN A,P_p instruction will sample the status of a PORT pin and feed in the correct logic level. With no external input connected, the IN A,P_p instruction inputs the previous output status.

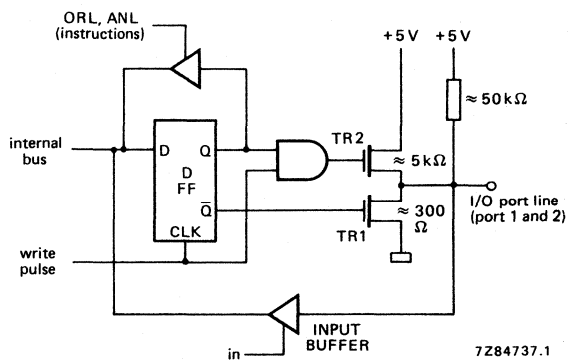


Fig. 18 Quasi-bidirectional port structure.

This PORT structure permits input and output information on the same pin and any mix of input and output lines on the same port. When a mix of inputs and outputs are present on one PORT, a PORT write will cause the strong internal pull-ups to turn on at all inputs. In this event, if a switch or some other low-impedance device is connected to an input, a PORT write (input logic 1) could cause excess current on internal lines. The recommended connection when inputs and outputs are mixed on one PORT is shown in Fig. 19. This bidirectional port structure combined with the MAB8041A logical AND and OR instructions provides an efficient method of handling single line inputs and outputs in an 8-bit processor.

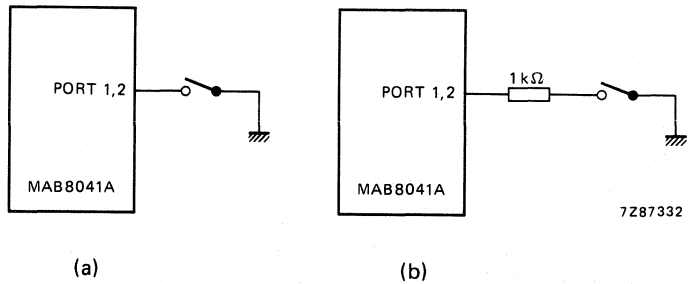


Fig. 19 Recommended method of PORT connection; (a) when all lines on the PORT are inputs; (b) when inputs and outputs on a PORT are mixed.

Power-down mode (see Fig. 20)

In the MAB8041A, power can be removed from all but the 64 x 8 data RAM array, for low power stand-by operation. In the power-down mode the contents of the data RAM can be maintained while drawing typically 10% to 15% of the normal operating supply voltage. V_{CC} serves as the + 5 V supply pin for the bulk of the circuitry, while the VDD pin supplies only the RAM array. In normal operation, both pins are at + 5 V. In standby, V_{CC} is at ground and only V_{DD} is maintained at + 5 V. Applying $\overline{\text{RESET}}$ to the processor through the $\overline{\text{RESET}}$ pin inhibits any access to the RAM by the processor and guarantees that the RAM cannot be inadvertently altered as power is removed from V_{CC}.

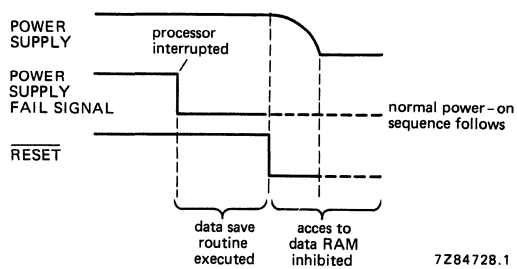


Fig. 20 Power down sequence.

The power-down sequence is as follows:

1. User-defined circuits detect imminent power failure. The detection signal must give adequate time for the MAB8041A to save necessary data before V_{CC} drops out of normal operating tolerance.
2. A 'power failure' signal interrupts the processor (via a timer overflow interrupt for example) and calls a power failure service routine.
3. The power failure routine saves all important data and machine status in the RAM. It may also initiate the transfer of a back-up supply to the V_{DD} pin and give an external indication that the power failure routine is complete.
4. A $\overline{\text{RESET}}$ signal, applied by external hardware, ensures that data is not altered as the power supply falls out of limits. The applied $\overline{\text{RESET}}$ must remain LOW until V_{CC} reaches ground potential.

Recovery from the power-down mode occurs as for any other power-on sequence. An external 1 μF capacitor connected to the $\overline{\text{RESET}}$ input provides the required initialization pulse.

INSTRUCTION SET (see Tables 5 and 6)

The MAB8041A instruction set consists of 93 one and two-byte instructions. The instruction set is opcode-compatible with the MAB8048 set except for the elimination of external program and data memory instructions and the addition of the data bus buffer instructions. It is extremely straightforward and program code efficiency is high.

- Over 70% of instructions are one byte long; the remainder are two-byte.
- Over 50% of instructions are executed in one machine cycle; the remainder require two including branch, immediate and I/O operations.

The instruction set efficiently manipulates the single-bit operations necessary for control functions. Special instructions allow port bits to be individually set or cleared. Also any accumulator bit can be directly tested via conditional branch instructions. Additional instructions simplify loop counters, table look-up routines and N-way branch routines. Arithmetic operations are handled in both binary and binary-coded decimal (BCD) notations for efficient interfacing of peripherals such as keyboards and displays.

The MAB8041A instruction set sub-divides into nine groups:

- Data moves
- Accumulator operations
- Flags
- Register operations
- Branch instructions
- Control
- Timer operations
- Subroutines
- Input/output instructions

The mnemonics and operational details of all these groups are given in Table 6.

INSTRUCTION SET (continued)

Data moves

All data transfers within the MAB8041A are controlled from the 8-bit accumulator. Data can be transferred between the 8 registers of each working register bank and the accumulator directly, with source or destination specified by 3 instruction bits. The other locations in the RAM array are addressed by either R0 or R1 of the active register bank. Transfers to and from the RAM require one cycle. Constants that are stored in program memory can be loaded directly into the accumulator or the eight working registers. Direct transfer of data is also possible between the accumulator and the internal timer/event counter, the status register (STS), or the program status word (PSW). Transfers to the STS register alter bits 4 to 7 only. Transfers to the PSW alter machine status accordingly and provide a means of restoring status after an interrupt or for altering the stack pointer if necessary.

Accumulator operations

It is possible to add (with or without carry) immediate data, data memory, or the working register contents to the accumulator. These sources can also be ANDed, ORed, or exclusive ORed to the accumulator. Data may be moved to or from the accumulator and working registers or data memory. The two values can be exchanged in a single operation.
The lower 4 bits of the accumulator can be exchanged with those of any of the internal RAM locations. This operation, together with an instruction which swaps the upper and lower 4-bit halves of the accumulator, facilitates the handling of BCD numbers and other 4-bit quantities. A decimal adjust instruction facilitates BCD arithmetic, which corrects the result of the binary addition of 2-digit BCD numbers in the accumulator.
The accumulator can be incremented, decremented, cleared, or complemented, and can be rotated left or right one bit at a time with or without carry.
Subtract operations are performed, under MAB8041A software control, using three single-byte, single-cycle instructions. A value can be subtracted from the accumulator using the following instructions:

- Complement the accumulator
- Add the value to the accumulator
- Complement the accumulator

Flags

There are four user accessible flags:

- Carry
- Auxiliary carry
- F0
- F1

The carry flag indicates overflow of the accumulator; the auxiliary carry indicates overflow between BCD digits, and is used during decimal adjust operations. Both these flags are part of the PSW and are stored in the program counter stack during subroutine calls.
F0 and F1 are general purpose flags, that can be cleared or complemented by MAB8041A instructions. F0 is accessible via the PSW and is also stored in the program counter stack. F1 indicates the condition of the A0 line, therefore care must be taken when setting or clearing it.

Register operations

As previously stated, the working registers can be accessed via the accumulator, or loaded with immediate data constants from program memory. Also, they can be incremented or decremented directly, or used as loop counters as explained in branch instructions.

Branch instructions

There are 17 jump instructions included in the MAB8041A instruction set. The unconditional jump instruction permits jumps anywhere in the 1 k words of program memory. All other jump instructions are limited to the current page (256 words) of program memory.

Conditional jump instructions can test the following inputs and machine flags:

- TEST 0 input pin
- TEST 1 input pin
- Input buffer full flag
- Output buffer full flag
- Timer flag
- Accumulator flag
- Accumulator bit
- Carry flag
- F0 flag
- F1 flag

The conditions tested by these instructions are the instantaneous values existing at the time the conditional jump instruction is executed. Thus, the jump on accumulator zero instruction tests the accumulator itself and not an intermediate flag.
The decrement register and jump if not zero (DJNZ) instruction combines decrement and branch operations in a single instruction, which can be used to implement a loop counter. Any of the 8 working registers can be designated as a counter by this instruction, which can effect a branch to any address on the current page of execution.
A special indirect jump instruction (JMPP @A) allows the program to be vectored to any one of several locations dependent on the contents of the accumulator, which point to a location in program memory that contains the jump address. This instruction can be used to vector to any one of several routines based on an ASCII character that has been loaded into the accumulator. Thus ASCII inputs can be used to initiate various routines.

Control

Control of the DMA interrupts, and selection of working register banks is achieved by six instructions. The MAB8041A provides two instructions for controlling the external microcontroller system. The IBF and OBF flags can be routed to PORT 2 to interrupt the external processor. DMA handshake signals can also be enabled by PORT 2 lines. The IBF interrupt can be enabled and disabled by two instructions. This interrupt is also automatically disabled after a $\overline{\text{RESET}}$ input or during an interrupt service routine. The working register bank switch instructions allows the programmer to substitute immediately a second 8-register bank for the one in use. This effectively provides either 16 working registers or the means for quickly saving the contents of the first 8 registers in response to an interrupt. It gives the user the option of switching register banks when an interrupt occurs. If the banks are switched, the original bank will automatically be restored when a return and restore status (RETR) instruction is performed at the end of the interrupt sevice routine.

INSTRUCTION SET (continued)

Timer

The 8-bit internal timer/event counter can be loaded or read via the accumulator with the event counter at rest or during counting.
The event counter can be started as a timer with an external clock applied to the TEST 1 input pin. The particular instruction executed determines which clock source is used; a single instruction stops the event counter independent of which clock source is used. Two instructions allow the timer interrupt to be enabled or disabled.

Subroutines

Subroutines are entered by executing a call instruction, available to any address in the 1 k-word program memory. Two separate return instructions determine whether or not status (upper 4 bits of the PSW) is restored on return from a subroutine.

Input/output instructions

Communication between the MAB8041A and the external microcontroller system is achieved via two 8-bit data bus buffer registers (DBBIN and DBBOUT) and an 8-bit status register (STS).
Data can be input from the DBBIN register to the accumulator, and output from the accumulator to the DBBOUT register.
The STS register contains four user-definable bits (ST4 to ST7) and four reserved status bits (IBF, OBF, F0 and F1) that are set from the accumulator. The MAB8041A peripheral interface has two 8-bit static I/O ports that can be loaded to and from the accumulator. Outputs are statically latched, but inputs are sampled at the time that an IN instruction is executed. Also immediate data from program memory can be ANDed and ORed directly to PORT 1 and PORT 2, the result remaining on the port. This permits masks stored in program memory to be used to set or reset individual bits on the I/O ports. Both ports are configured to allow input on a given pin by the first writing a logic 1 to it.

Instruction set description

The following section provides a detailed description of each MAB8041A peripheral interface instruction and illustrates the method in which they are used.
The symbols and abbreviations used in the instruction set, Table 6, are shown in Table 5.

Table 5 Symbols and definitions used in Table 6.

symbol	definition description
A	accumulator
C	carry flag
DBBIN	data bus buffer input
DBBOUT	data bus buffer output
F0, F1	flag 0, flag 1 (C/D flag)
I	interrupt
P	'in page' operation designation
PC	program counter
P_p	port designation (p = 1, 2 or 4-7)
PSW	program status word
Rr	register designation (r = 0, 1 or 0-7)
SP	stack pointer
STS	status register
T	timer
TF	timer flag
T0, T1	test 0 and 1 inputs
#	immediate data prefix
@	indirect address prefix
(X)	contents of X
(())	double parenthesis show the effect of @, that is, @R0 is shown as ((R0)).

INSTRUCTION SET (continued)

Table 6 Instruction set

	mnemonic		description	bytes	cycles
ACCUMULATOR	ADD	A,Rr	Add register to A	1	1
	ADD	A,@Rr	Add data memory to A	1	1
	ADD	A,#data	Add immediate to A	2	2
	ADDC	A,Rr	Add register to A with carry	1	1
	ADDC	A,@Rr	Add data memory to A with carry	1	1
	ADDC	A,#data	Add immediate to A with carry	2	2
	ANL	A,Rr	AND register to A	1	1
	ANL	A,@Rr	AND data memory to A	1	1
	ANL	A,#data	AND immediate to A	2	2
	ORL	A,Rr	OR register to A	1	1
	ORL	A,@Rr	OR data memory to A	1	1
	ORL	A,#data	OR immediate to A	2	2
	XRL	A,Rr	Exclusive OR register to A	1	1
	XRL	A,@Rr	Exclusive OR data memory to A	1	1
	XRL	A,#data	Exclusive OR immediate to A	2	2
	INC	A	Increment A	1	1
	DEC	A	Decrement A	1	1
	CLR	A	Clear A	1	1
	CPL	A	Complement A	1	1
	DA	A	Decimal adjust A	1	1
	SWAP	A	Swap nibbles of A	1	1
	RL	A	Rotate A left	1	1
	RLC	A	Rotate A left through carry	1	1
	RR	A	Rotate A right	1	1
	RRC	A	Rotate A right through carry	1	1
CONTROL	EN	DMA	Enable DMA handshake lines	1	1
	EN	I	Enable IBF interrupt	1	1
	DIS	I	Disable IBF interrupt	1	1
	EN	FLAGS	Enable master interrupts	1	1
	SEL	RB0	Select register bank 0	1	1
	SEL	RB1	Select register bank 1	1	1
	NOP		No operation	1	1

	mnemonic		description	bytes	cycles
INPUT/OUTPUT	IN	A,P_p	Input port to A	1	2
	OUTL	P_p,A	Output A to port	1	2
	ANL	P_p,#data	AND immediate to port	2	2
	ORL	P_p,#data	OR immediate to port	2	2
	IN	A,DBB	Input DBB to A, clear IBF	1	1
	OUT	DBB,A	Output A to DBB, set OBF	1	1
	MOV	STS,A	A4 to A7 to bits 4 to 7 of status	1	1
	MOVD	A,P_p	Input expander port to A	1	2
	MOVD	P_p,A	Output A to expander port	1	2
	ANLD	P_p,A	AND A to expander port	1	2
	ORLD	P_p,A	OR A to expander port	1	2
DATA MOVES	MOV	A,Rr	Move register to A	1	1
	MOV	A,@Rr	Move data memory to A	1	1
	MOV	A,#data	Move immediate to A	2	2
	MOV	Rr,A	Move A to register	1	1
	MOV	@Rr,A	Move A to data memory	1	1
	MOV	Rr,#data	Move immediate to register	2	2
	MOV	@Rr,# data	Move immediate to data memory	2	2
	MOV	A,PSW	Move PSW to A	1	1
	MOV	PSW,A	Move A to PSW	1	1
	XCH	A,Rr	Exchange A and registers	1	1
	XCH	A,@Rr	Exchange A and data memory	1	1
	XCHD	A,@Rr	Exchange digit of A and register	1	1
	MOVP	A,@A	Move to A from current page	1	2
	MOVP3	A,@A	Move to A from page 3	1	2
TIMER/COUNTER	MOV	A,T	Read timer/counter	1	1
	MOV	T,A	Load timer/counter	1	1
	STRT	T	Start timer	1	1
	STRT	CNT	Start counter	1	1
	STOP	TCNT	Stop timer/counter	1	1
	EN	TCNTI	Enable timer/counter interrupt	1	1
	DIS	TCNTI	Disable timer/counter interrupt	1	1

INSTRUCTION SET (continued)

	mnemonic		description	bytes	cycles
REG.	INC	Rr	Increment register	1	1
	INC	@Rr	Increment data memory	1	1
	DEC	Rr	Decrement register	1	1
SUBR.	CALL	addr	Jump to subroutine	2	2
	RET		Return	1	2
	RETR		Return and restore status	1	2
FLAGS	CLR C		Clear carry	1	1
	CPL C		Complement carry	1	1
	CLR F0		Clear flag F0	1	1
	CPL F0		Complement flag F0	1	1
	CLR F1		Clear flag F1	1	1
	CPL F1		Complement flag F1	1	1
BRANCH	JMP	addr	Jump unconditional	2	2
	JMPP	@A	Jump indirect	1	2
	JC	addr	Jump on carry = 1	2	2
	JNC	addr	Jump on carry = 0	2	2
	JZ	addr	Jump on A zero	2	2
	JNZ	addr	Jump on A not zero	2	2
	JT0	addr	Jump on T0 = 1	2	2
	JNT0	addr	Jump on T0 = 0	2	2
	JT1	addr	Jump on T1 = 1	2	2
	JNT1	addr	Jump on T1 = 0	2	2
	JF0	addr	Jump on flag F0 = 1	2	2
	JF1	addr	Jump on flag F1 = 1	2	2
	JTF	addr	Jump on timer flag = 1	2	2
	JNIBF	addr	Jump on IBF flag = 0	2	2
	JOBF	addr	Jump on OBF flag = 1	2	2
	JBb	addr	Jump on accumulator bit	2	2
	DJNZ	Rr,addr	Decrement register and jump on non-zero	2	2

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Input voltage with respect to V_{SS}			
except input EA	V_I	−0,5 to + 7	V
input EA	V_I	−0,5 to + 15	V
D.C. current into any input or output	$\pm I_I, \pm I_O$	max. 10	mA
Total power dissipation	P_{tot}	max. 1,5	W
Storage temperature range	T_{stg}	−65 to + 150	°C
Operating ambient temperature range	T_{amb}	0 to + 70	°C

D.C. CHARACTERISTICS

V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	typ.	max.	unit
Supply voltage	$V_{CC} = V_{DD}$	4,5	5	5,5	V
Supply current	I_{DD}	–	–	15	mA
Supply current (total)	$I_{CC} + I_{DD}$	–	–	125	mA
Inputs					
Input voltage LOW all inputs except XTAL 1, XTAL 2, $\overline{RESET}$	V_{IL}	–0,5	–	0,8	V
Input voltage LOW XTAL 1, XTAL 2, $\overline{RESET}$	V_{IL1}	–0,5	–	0,6	V
Input voltage HIGH all inputs except XTAL 1, XTAL 2, $\overline{RESET}$	V_{IH}	2,2	–	V_{CC}	V
Input voltage HIGH XTAL 1, XTAL 2, $\overline{RESET}$	V_{IH1}	3,8	–	V_{CC}	V
Outputs					
Output voltage LOW (DB0 to DB7) at I_{OL} = 2 mA	V_{OL}	–	–	0,45	V
Output voltage LOW (P10 to P17, P20 to P27, SYNC) at I_{OL1} = 1,6 mA	V_{OL1}	–	–	0,45	V
Output voltage LOW (PROG) at I_{OL2} = 1 mA	V_{OL2}	–	–	0,45	V
Output voltage HIGH (DB0 to DB7) at $-I_{OH}$ = 400 μA	V_{OH}	2,4	–	–	V
Output voltage HIGH (all other outputs) at $-I_{OH1}$ = 50 μA	V_{OH1}	2,4	–	–	V
Input leakage current (T0, T1, $\overline{RD}$, $\overline{WR}$, $\overline{CS}$, A0, EA) at $V_{SS} \leqslant V_I \leqslant V_{CC}$	$\pm I_{IL}$	–	–	10	μA
Output leakage current (DB0 to DB7; high impedance) at V_{SS} + 0,45 V $\leqslant V_I \leqslant V_{CC}$	$\pm I_{OZ}$	–	–	10	μA
LOW input load current ((P10 to P17, P20 to P27)) at V_{IL} = 0,8 V	I_{LI}	–	–	0,5	mA
LOW input load current ($\overline{RESET}$, $\overline{SS}$) at V_{IL} = 0,8 V	I_{LI1}	–	–	0,2	mA

A.C. CHARACTERISTICS

$V_{CC} = V_{DD} = 5\ V \pm 10\%$; $V_{SS} = 0\ V$; $T_{amb} = 0$ to $+70\ ^{o}C$ unless otherwise specified

parameter	symbol	min.	typ.	max.	unit
DBB read cycle (see Fig. 22)					
$\overline{CS}$, A0 set-up time $\overline{RD}1$	t_{AR}	0	–	–	ns
$\overline{CS}$, A0 hold time after $\overline{RD}1$	t_{RA}	0	–	–	ns
$\overline{RD}$ pulse duration	t_{RR}	250	–	–	ns
$\overline{CS}$, A0 to data out delay time at C_L = 150 pF	t_{AD}	–	–	225	ns
$\overline{RD}1$ to data out delay time at C_L = 150 pF	t_{RD}	–	–	225	ns
$\overline{RD}1$ to data float delay time	t_{DF}	–	–	100	ns
Read cycle time (6,0 MHz crystal)	t_{CY}	2,5	–	15	μs
Read cycle time (3,6 MHz crystal)	t_{CY}	4,17	–	15	μs
DBB write cycle (see Fig. 23)					
$\overline{CS}$, A0 set-up time to $\overline{WR}1$	t_{AW}	0	–	–	ns
$\overline{CS}$, A0 hold time after $\overline{WR}1$	t_{WA}	0	–	–	ns
$\overline{WR}$ pulse duration	t_{WW}	250	–	–	ns
Data set-up time to $\overline{WR}1$	t_{DW}	150	–	–	ns
Data hold time after $\overline{WR}1$	t_{WD}	0	–	–	ns
PORT 2 timing (see Fig. 24)					
Port control set-up time before falling edge of PROG	t_{CP}	110	–	–	ns
Port control set-up time after falling edge of PROG	t_{PC}	100	–	–	ns
PROG to time P2 input must be valid	t_{PR}	–	–	810	ns
Input data hold time	t_{PF}	0	–	150	ns
Output data set-up time	t_{DP}	250	–	–	ns
Output data hold time	t_{PD}	65	–	–	ns
PROG pulse duration	t_{PP}	1200	–	–	ns
DMA TIMING (see Fig. 25)					
$\overline{DACK}$ to $\overline{WR}$ or $\overline{RD}$	t_{ACC}	0	–	–	ns
$\overline{WR}$ or $\overline{RD}$ to $\overline{DACK}$	t_{CAC}	0	–	–	ns
$\overline{DACK}$ to data valid at C_L = 150 pF	t_{ACD}	–	–	225	ns
$\overline{WR}$ or $\overline{RD}$ to DRQ cleared	t_{CRQ}	–	–	200	ns

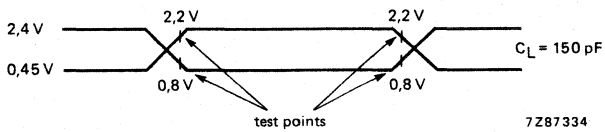


Fig. 21 Input and output waveforms for a.c. tests.

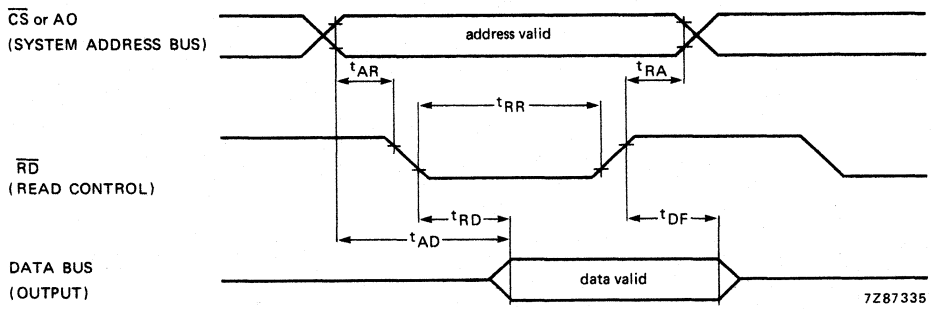


Fig. 22 Read from data bus buffer register.

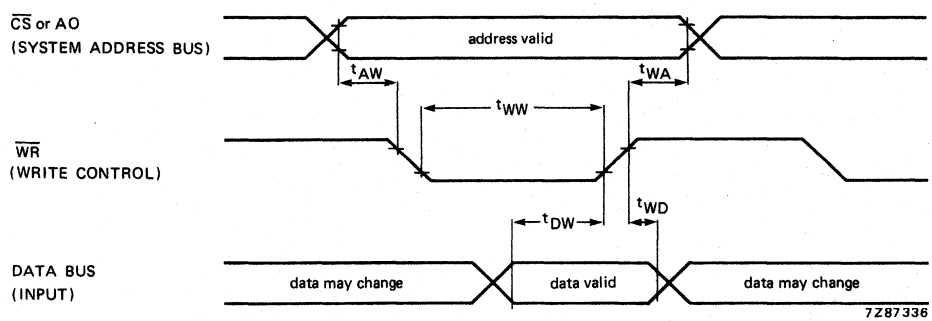


Fig. 23 Write to data bus buffer register.

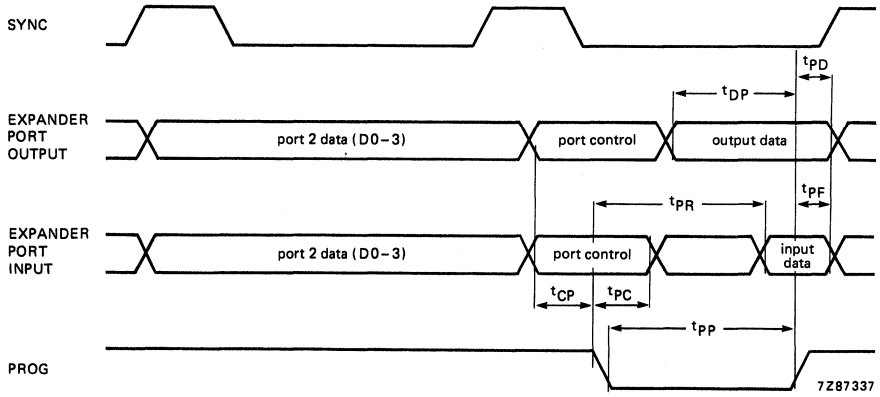


Fig. 24 Port 2 timing.

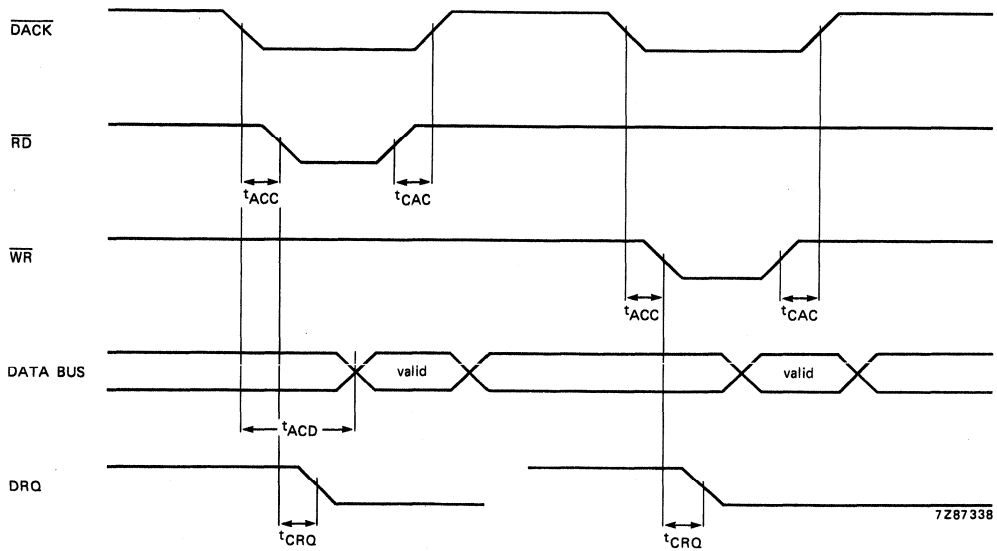


Fig. 25 DMA timing.

SINGLE-CHIP 8-BIT MICROCONTROLLER

DESCRIPTION

The MAB84XX family of microcontrollers is fabricated in NMOS. The family consists of 8 devices:

- MAB8400 – 128 RAM bytes, external program memory
- MAB8401 – like 8400 but with 8-bit LED-driver (10 mA), emulation of MAB/F8422/42* possible
- MAB/F8411 – 1K ROM/ 64 RAM bytes plus 8-bit LED-driver ←
- MAB/F8421 – 2K ROM/ 64 RAM bytes plus 8-bit LED-driver
- MAB/F8441 – 4K ROM/128 RAM bytes plus 8-bit LED-driver
- MAB/F8461 – 6K ROM/128 RAM bytes plus 8-bit LED-driver

Each version has 20 quasi-bidirectional I/O port lines, one serial I/O line, one single-level vectored interrupt, an 8-bit timer event counter and on-board clock oscillator and clock circuits. Two 20-pin versions, MAB/F8422 and MAB/F8442* are also available.

This microcontroller family is designed to be an efficient controller as well as an arithmetic processor. The instruction set is based on that of the MAB8048. The microcontrollers have extensive bit handling abilities and facilities for both binary and BCD arithmetic.

For detailed information see the "Users manual Single-chip microcomputers" (supplied upon request).

* See data sheet on MAB/F8422/42.

Features

- 8-bit: CPU, ROM, RAM and I/O in a single 28-lead DIL package
- 1K, 2K, 4K or 6K ROM bytes plus a ROM-less version
- 64 or 128 RAM bytes
- 20 quasi-bidirectional I/O port lines
- Two testable inputs: one of which can be used to detect zero cross-over, the other is also the external interrupt input
- Single level vectored interrupts: external, timer/event counter, serial I/O
- Serial I/O that can be used in single or multi-master systems (serial I/O data via an existing port line and clock via a dedicated line)
- 8-bit programmable timer/event counter
- Internal oscillator, generated with inductor, crystal, ceramic resonator or external source
- Over 80 instructions (based on MAB8048) all of 1 or 2 cycles
- Single 5 V power supply (± 10%)
- Operating temperature ranges:

0 to + 70 °C	MAB84XX	family
–40 to + 85 °C	MAF84XX	family
–40 to + 110 °C	MAF84AXX	family

PACKAGE OUTLINES

MAB8400/01B: 28-lead 'Piggy-back' package (with up to 28-pin EPROM on top)
MAB8400/01WP: 68-lead plastic leaded chip-carrier (PLCC) (SOT-188A)
MAB/F8411/21/41/61P: 28-lead DIL; plastic (SOT-117D)
MAB8411/21/41T: 28-lead mini-pack; plastic (SO-28; SOT-136A).

PINNING

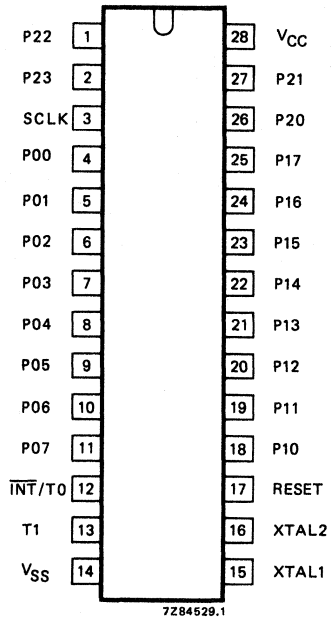
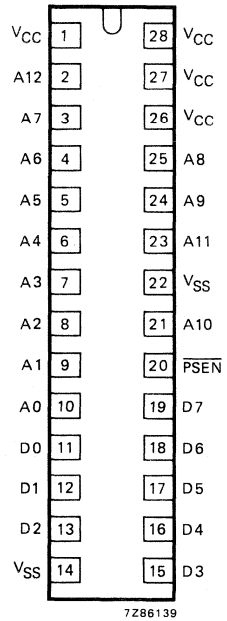


Fig. 1 Pinning diagram for mask-programmable devices MAB8411, MAB8421, MAB8441, MAB8461 and for MAB8400 and MAB8401 'Piggy-back' version bottom pinning (for top pinning see Fig. 2).

PINNING DESIGNATION

V_{SS}	14	**Ground**
V_{CC}	28	**Power supply,** + 5 V
P00 – P07	4 – 11	**Port 0,** 8-bit quasi-bidirectional I/O port
P10 – P17	18 – 25	**Port 1,** 8-bit quasi-bidirectional I/O port
P20 – P23	26, 27, 1, 2	**Port 2,** 4-bit quasi-bidirectional I/O port; P23 is the serial data I/O in serial I/O mode
SCLK	3	Bidirectional clock for serial I/O
$\overline{INT}$/T0	12	External interrupt input (sensitive to a negative-going edge min LOW > 7 clock pulses, min HIGH > 4 clock pulses), testable using the JT0 or JNT0 instructions.
T1	13	Input pin, testable using the JT1 or JNT1 instructions. It can be designated as event counter input using the STRT CNT instruction. It can also be used to detect zero cross-over of slowly moving a.c. inputs.
→ RESET	17	Input to initialize the processor (active HIGH).
→ XTAL1	15	Connection to timing component (crystal) that determines the frequency of the internal oscillator. It is also the input for an external clock source.
→ XTAL2	16	Connection to other side of the timing component.

MAB8400B/01B (top pinning)



PIN DESIGNATION

designation	pin	function
V_{SS}	14, 22	**Ground**
V_{CC}	1, 26–28	**Power supply,** + 5 V
A0–A12	10–3, 25, 24, 21, 23, 2	Address outputs ←
D0–D7	11–13, 15–19	Data inputs
$\overline{PSEN}$	20	Program store enable

Fig. 2(a) Pinning diagram for MAB8400/01B 'Piggy-back' version top pinning (for bottom pinning see Fig. 1); to access a 2732 or 2764 EPROM.

Note

Access times for ROMS/EPROMS to be below 1 μs.

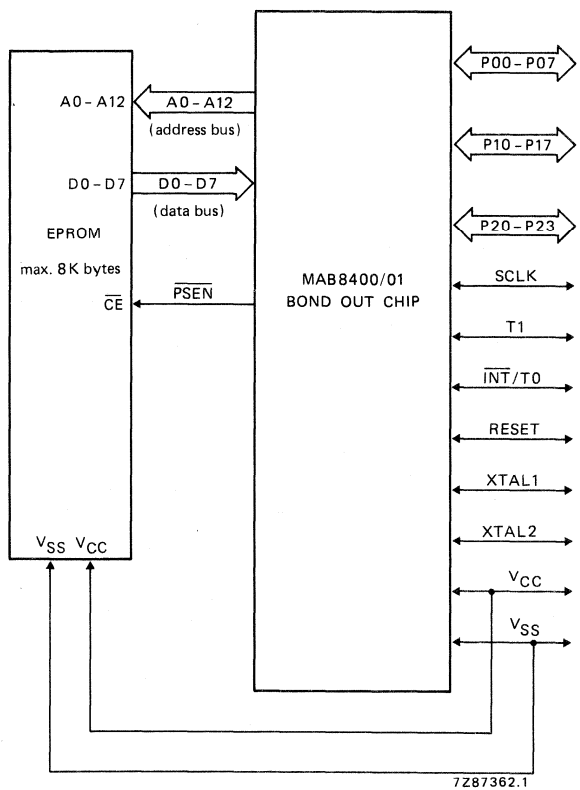


Fig. 2(b) Connection of EPROM to 'Piggy-back' package MAB8400/01B.

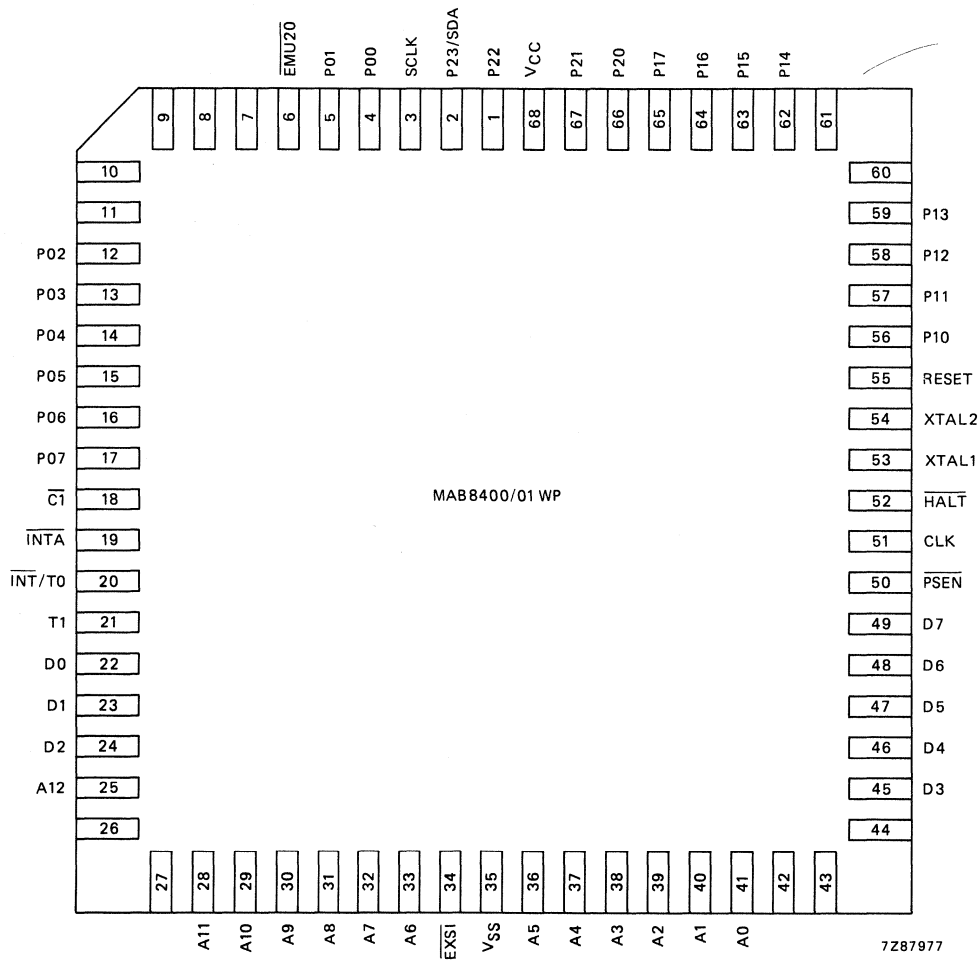
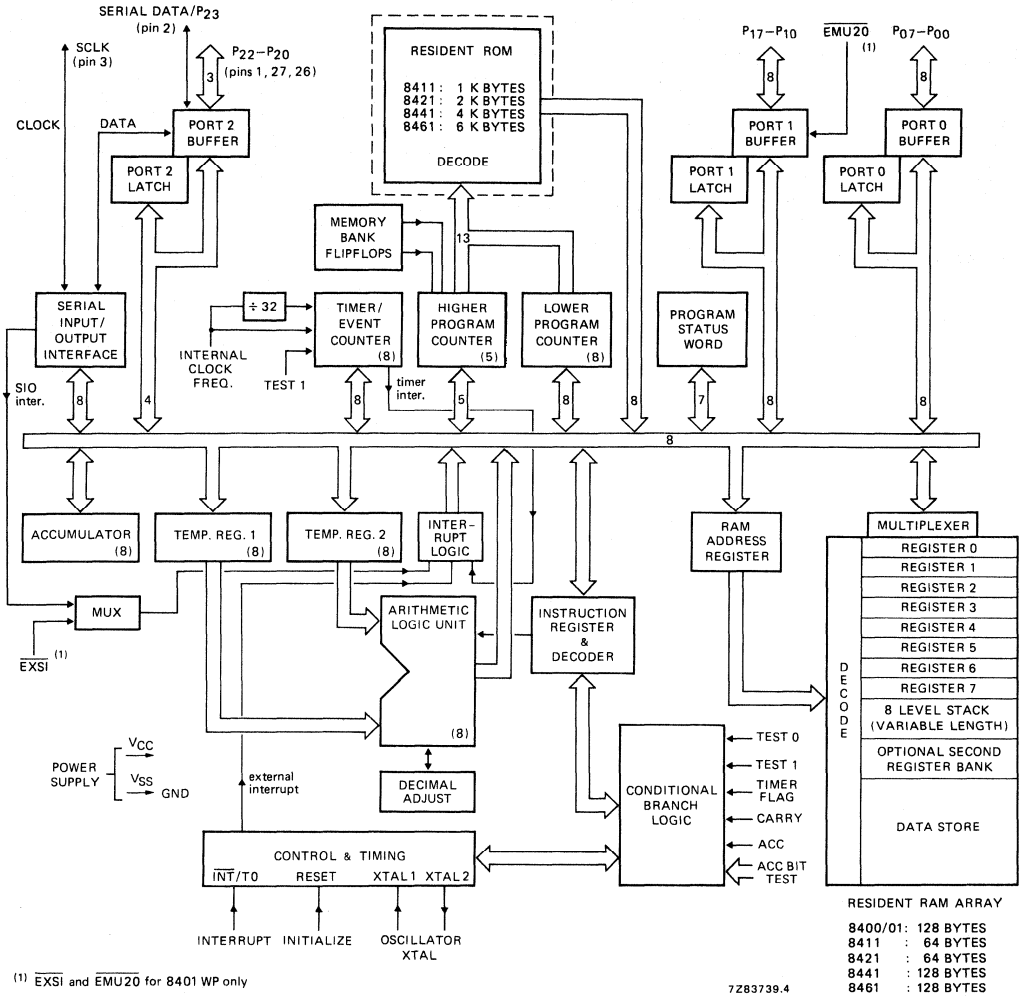


Fig. 3 Pad diagram for PLCC.

CHIP CARRIER DESIGNATION

designation	pad no.	function
V_{SS}	35	**Ground**
V_{CC}	68	**Power supply,** + 5 V
P00–P07	4–5, 12–17	**Port 0,** 8-bit quasi-bidirectional I/O port
P10–P17	56–59, 62–65	**Port 1,** 8-bit quasi-bidirectional I/O port
P20–P22	66, 67, 1	**Port 2,** 4-bit quasi-bidirectional I/O port; P23 is the serial data I/O in series I/O mode
P23/SDA	2	
SCLK	3	Bidirectional clock for serial I/O
$\overline{INT}$/T0	20	External interrupt input (sensitive to a negative-going edge), testable using the JT0 or JNT0 instructions

T1	21	Input pin, testable using the JT1 or JNT1 instructions. It can be designated as event counter input using the STRT CNT inctruction. It can also be used to detect zero cross-over of slowly moving a.c. inputs.
RESET	55	Input to initialize the processor (active HIGH)
XTAL1	53	Connection to timing component (e.g. crystal) that determines the frequency of the internal oscillator. It is also the input for an external clock source.
XTAL2	54	Connection to other side of the timing component
$\overline{EXSI}$	34	External serial I/O interrupt (active-LOW) for emulation of MAB/F8422/42 (8400 back-bias)
A0–A12	41–36, 33–28 25	Program memory address outputs (active HIGH); A0 = LSB, A12 = MSB. Address output change after begin Phi3 of TS8.
D0–D7	22–24, 45–49	Data input lines (active HIGH) used for reading external program memory. D0 = LSB, D7 = MSB.
CLK	51	Clock output buffered from XTAL2. On the positive-going edge the (internal) Phi clock goes HIGH.
$\overline{PSEN}$	50	Program store enable. This signal is used for enabling the external EPROM (e.g. on the 'Piggy-back' version). For emulation, it enables the emulation memory and it indicates machine cycles. Active LOW during TS9, TS10 of each machine cycle and TS1 of the following machine cycle.
$\overline{C1}$	18	Cycle 1 indication output (active LOW). During emulation, this signal indicates the opcode fetch cycle (useful for external instruction decoding, real-time trace). Active from start of TS10 of the cycle preceding cycle 1, until the start of TS10 of cycle 1.
$\overline{HALT}$	52	Halt input (active LOW). If activated, the current instruction is finished and the microcontroller stops execution (HALT mode). The next program counter address is available on the address bus. Program counter and timer/event counter are no longer updated. The serial I/O finishes the current transmit/receive action and goes into the idle state. Interrupts are *not* sampled in the HALT mode, they are only sampled when the microcontroller is running. Interrupt routines can be single-stepped as a normal program.
$\overline{INTA}$	19	Interrupt acknowledge output (active LOW). It indicates any interrupt acception. Active from start of TS8 of the interrupted cycle, until start of TS7 of the second cycle of the (internally forced 'CALL vector address' instruction. During $\overline{INTA}$ active, the address bus shows the address that has been saved in the stack (return address); the C1 output indicates opcode fetch cycles as if a user CALL was executed.
$\overline{EMU20}$	6	Emulate 20-pin version MAB/F8422/42 (active-LOW) (8400 not connected)



(1) EXSI and EMU20 for 8401 WP only

Fig. 4(a) Block diagram of the MAB84XX family.

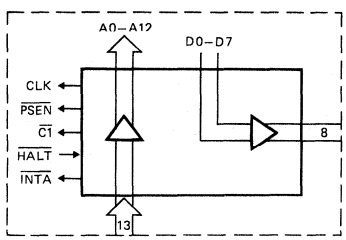


Fig. 4(b) Replacement for dotted part in Fig. 4(a) for the MAB8401WP bond-out version.

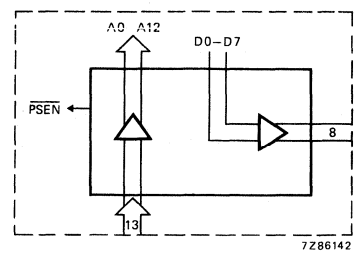


Fig. 4(c) Replacement of dotted part in Fig. 4(a) for the MAB8400B/01B 'Piggy-back' version.

FUNCTIONAL DESCRIPTION (for more detail see Microcontroller Users Manual)

Bond-out version MAB8400WP/01WP

The bond-out version is a microcontroller that contains no on-board ROM, but has all address and data lines brought out to access an external ROM or EPROM. So, this version has more pins than the standard microcontrollers with on-board ROM. It has all the features of the other members of the MAB84XX family, including emulation facilities for the MAB/F8422/42 (20-pin version). It can address 8K bytes of ROM. The RAM has 128 bytes.

Piggy-back version MAB8400B/01B

The Piggy-back version is a special package that has standard pinning to the bottom which facilitates insertion as a mask-programmed device. An EPROM is mounted on top in an additional socket. Thus, the total package height is greater than the standard DIL package. Emulation of the 8422/42 is not possible.

Program and data memory

The program memory (ROM) is mask-programmed at our factory. Because the MAB84XX family offers a range of ROM capacities to suit the application, ROM expansion is not required. Figure 5 shows the program memory map. Program memory is arranged in banks of 2K bytes, that are selected by SEL MB instructions.

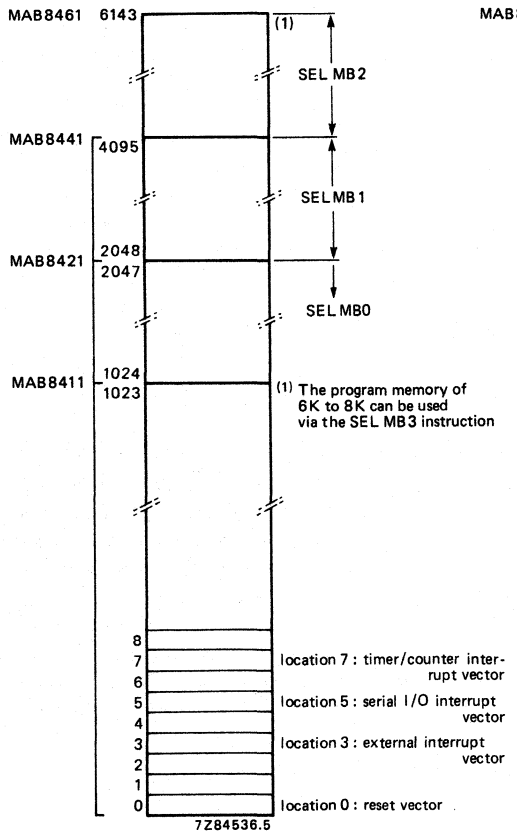


Fig. 5 The program memory map.

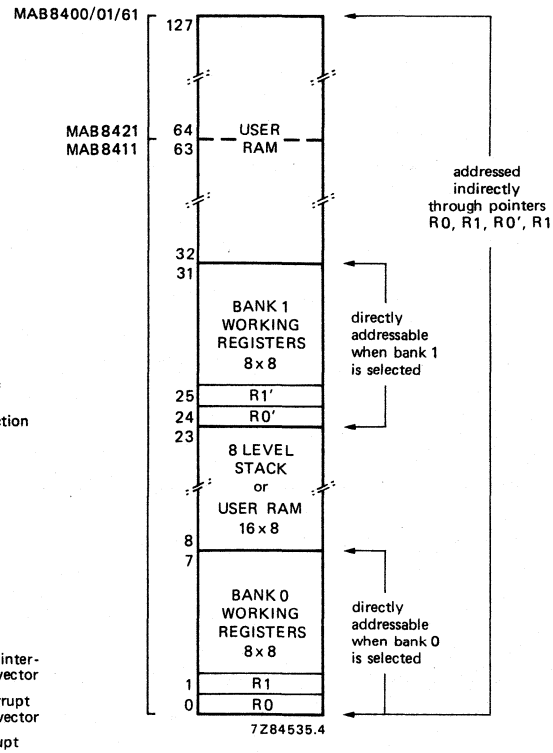


Fig. 6 The data memory map.

FUNCTIONAL DESCRIPTION (continued)

The data memory (RAM) consists of 64 or 128 bytes (8-bit words). All locations are indirectly addressable using RAM pointer registers and up to 16 designated location can be addressed directly. The memory also includes an 8-level program counter stack addressed by a 3-bit stack pointer. Figure 6 shows the data memory map.

On-chip peripheral functions

In addition to the CPU and memories, an interrupt system, I/O facilities, and an 8-bit timer/event counter are integrated on-chip to assist the CPU in repetitions, complicated or time-critical tasks. The I/O facilities include the I/O pins, parallel ports and a serial I/O port, consisting of a data line SDA shared with a parallel port line (P23), and a dedicated clock line SCLK.

I/O facilities

The MAB84XX family has 23 I/O lines arranged as:

- Two parallel ports of 8 lines (P00–P07, P10–P17);
- A parallel port of 4 lines (P20–P23)
- A serial I/O consisting of a data line shared with a parallel port line (P23) and a separate clock line SCLK;
- An external interrupt and test input $\overline{\text{INT}}$/T0, which when used as a test input can be tested by the conditional jump instructions JT0 or JNT0;
- A test input T1, which can alter program sequences when tested by conditional jump instructions JT1 or JNT1. T1 can also be used as an input to the timer/event counter or to detect zero cross-over of slowly moving a.c. signals.

All parallel port lines are available in three optional output configurations (except P23 – option 1 only):

- Option 1; open drain output without pull-up transistor (Fig. 7(a))
- Option 2; open drain output with pull-up transistor (Fig. 7(b))
- Option 3; push-pull output with pull-up transistor (Fig. 7(c))

If the inputs and outputs on a port are mixed (mixed-mode), the inputs should be options 1 or 2 but not option 3. This prevents cross-currents via TR2 and an external connection to ground, while switching the output on the same port and in parallel, masking the inputs with logic '1' s.

The MAB84XX family serial I/O interface has been designed to eliminate the heavy processing load imposed upon a normal microcontroller performing serial data transfer. Whereas a normal microcontroller must regularly monitor the serial data bus for the presence of data, the serial I/O interface detects, receives and converts the serial data stream into a parallel format without interrupting the execution of the current program. An interrupt is sent to the microcontroller only when a complete byte is received. Then, the microcontroller reads the data byte in one instruction. Likewise, for transmission, the serial I/O interface performs parallel to serial conversion and subsequent serial output of the data and the microcontroller is only interrupted in the execution of its programmed tasks when a complete byte has been transmitted. The design of the serial I/O interface allows any number of MAB84XX family devices and peripheral circuits with I^2C bus compatibility to be interconnected by the two-line serial bus. This is achieved by allocating a specific 7-bit address to each device and ensuring that a device reacts only to a message preceded by its own address or the 'general call' address.

Address recognition is performed by the interface hardware so that the microcontroller need only be interrupted when a valid address is received. This saves significant processing time and memory space compared to a conventional microcontroller with a software serial interface. When the address facility is not required, for instance in a system with only two microcontrollers, direct data transfer is possible. In multi-master systems, an automatically invoked arbitration procedure prevents two or more devices transmitting simultaneously.

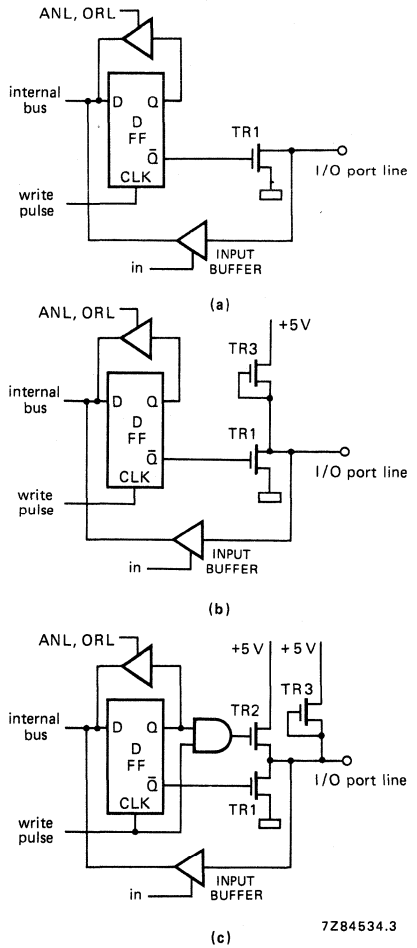


Fig. 7 Quasi-bidirectional I/O interface with (a) open drain output without pull-up transistor, (b) open drain output with pull-up transistor, (c) push-pull output with pull-up transistor.

Serial I/O interface

Figure 8 shows the serial I/O interface. The clock line of the serial bus has exclusive use of pin 3 (SCLK) while the data line shares pin 2 (serial data) with the I/O line P23 of port 2. When the serial I/O is enabled, P23 is disabled as a parallel port line (P23 and SCLK only open drain).

The microcontroller and interface communicate via the internal microcontroller bus and the Serial Interrupt Request line. Data and information controlling the operation of the interface are stored in four registers:

- data shift register S0,
- serial I/O interface status word S1,
- serial clock control word S2,
- address register.

FUNCTIONAL DESCRIPTION (continued)

Serial I/O interface (continued)

Data shift register S0

S0 is the shift register that converts serial data to parallel format and vice versa. A pending interrupt is generated only after a complete byte has been transmitted, or after a complete data byte, specific or general call address has been received. The most significant bit is transmitted first.

Status word S1

S1 provides information about the state of the interface and stores interface control information from the microcontroller. The four most significant bits are common to both read and write instructions, with a separate 4 read-only control bits and 4 write-only interface status bits.

MST and TRX

These bits determine the operating mode of the serial I/O interface (Table 2).

Table 1 Operating modes of the serial I/O interface.

MST	TRX	mode
0	0	slave receiver
1	0	master receiver
0	1	slave transmitter
1	1	master transmitter

BB: Bus Busy

This bit indicates the status of the bus.

PIN: Pending Interrupt Not

PIN = '0' indicates that there is an interrupt pending. This causes a Serial Interrupt Request when the serial interrupt mechanism is enabled.

ESO: Enable Serial Output

The ESO flag enables/disables the serial I/O interface: ESO = '1' enables
ESO = '0' disables

BC0, BC1 and BC2

These bits indicate the number of bits received or transmitted in a serial data stream.

Bits ES0, BC0, BC1 and BC2 can only be written via software.

AL: Arbitration Lost

The AL flag is set via the hardware when the serial I/O interface, as a master transmitter, loses the bus arbitration procedure.

ASS: Addressed As Slave

This flag is set via the hardware when the interface detects either its own address or the 'general call' address as the first byte of a transfer and if the interface has been programmed to operate in the address recognition mode.

AD0: Address Zero

This flag is set via the hardware after the general call address is detected when the interface is operating in the address recognition mode.

LRB: Last Received Bit

This contains either the last data bit received or, for a transmitting device in the acknowledge mode, the acknowledge from the receiving device.

Bits AL, AAS, AD0 and LRB can only be read via software.

Clock control register S2

Bits 0 to 4 of S2 are used to set the frequency of the serial clock signal. When a 4,43 MHz crystal is used, the frequency of the serial clock can be varied between 100 kHz and 720 Hz. An asymmetrical clock with a HIGH to LOW ratio of 3 to 1 is produced by setting bit 5. The asymmetrical clock allows a microcontroller more time per clock period for sampling the data line, making the timing of this action less critical. Bit 6 is used to activate the acknowledge mode of the serial I/O. S2 is a write-only register.

Address register

The address register contains the 7-bit address back-up latches and the bit (ALS) used to enable/disable the address recognition mode. Only when ES0 = 0 can the address register be written using the MOV S0,A and MOV S0,#data instructions.

Serial I/O interrupt logic

The interrupt logic is enabled by the EN SI instruction and disabled by DIS SI. When the interrupt logic is enabled, a pending interrupt results in a serial I/O interrupt to the controller, causing a jump to location 5 in the ROM. When the logic is disabled, the presence of an interrupt is still indicated by the PIN bit in register S1. Therefore, an interrupt can still be serviced but a vectored interrupt will not occur.

Interrupt system

External events and real-time on-chip peripherals require servicing by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution, three single-level nested interrupts are provided.

Each interrupt vectors to a separate location in the program memory for its service program. Each source can be individually enabled or disabled. When more than one interrupt occurs simultaneously, their priority will be: (1) external, (2) serial I/O and, (3) timer/event counter. An additional external interrupt can be created using the timer/event counter interrupt.

FUNCTIONAL DESCRIPTION (continued)

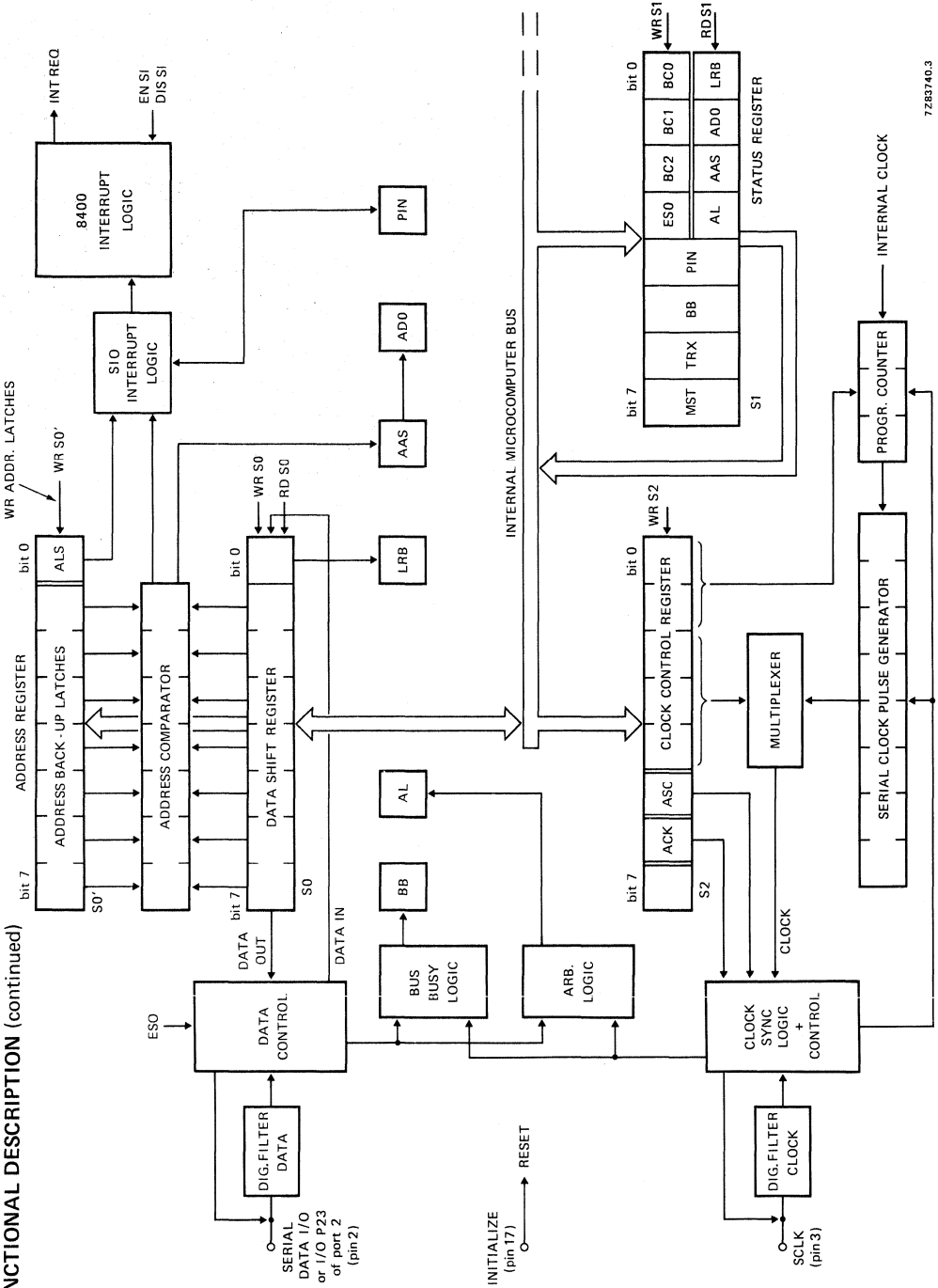


Fig. 8 The serial I/O interface.

Test input T1

The T1 input line can be used as:

- a test input for branch instructions,
- an input for zero voltage cross-over detection,
- an external input to the event counter.

An internal pull-up transistor is provided as a ROM mask option. This is useful when the input is from a switch or standard TTL output.

When T1 is used as a test input, the JT1 or JNT1 instructions test for a HIGH or a LOW respectively. The T1 input has a self-biasing circuit that can detect when an a.c. signal crosses zero (within ± 100 mV when coupled through a 1 μF capacitor)*. The maximum input voltage is 3 V (peak-to-peak), the maximum frequency is 1 kHz. Zero cross-over detection used in conjunction with the timer/event counter interrupt is useful in thyristor control of power equipment.

The operation of T1 as an input to the timer/event counter is described under the heading Timer/event counter.

High current outputs

Four pins are provided that can sink currents (typical values):

– P23 (serial data), pin 2	5 mA at 0,45 V (open drain),
– SCLK, pin 3	5 mA at 0,45 V (open drain),
– P10, pin 18**	7 mA at 2,5 V,
– P11, pin 19**	7 mA at 2,5 V,
– For MAB84X1 P10–P17▲	10 mA at 1 V (X = 0, 1, 2, 4 or 6) ←

* Typical design values, not guaranteed.

** P10 and P11 may be connected in parallel (if their logic outputs are always the same) to give 14 mA at 2,5 V.

▲ P10 to P17 may be connected in parallel if their logic outputs are always the same.

FUNCTIONAL DESCRIPTION (continued)

Timer/event counter

An 8-bit binary up-counter is provided. This can count external events, machine cycles divided by 32, or machine cycles directly. When used as a timer, the input to the counter is either the overflow or input of a 5-bit prescaler. When used as an event counter, LOW to HIGH transitions on T1 (pin 13) are counted. The maximum rate at which the counter may be incremented is once every machine cycle (200 kHz for a 5 μs machine cycle). Figure 9 illustrates the timer/event counter.

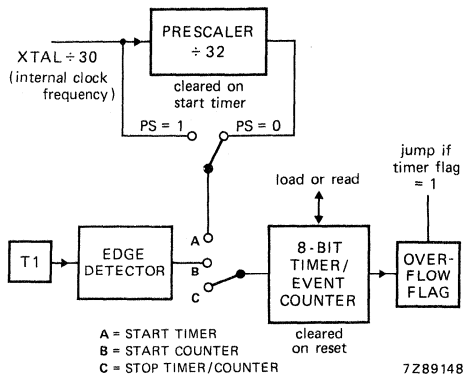


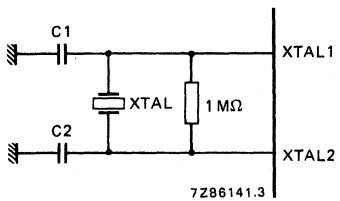
Fig. 9 The timer/event counter.

Differences between the MAB8021 and MAB8048 microcontrollers, and the MAB84XX family.

	8021	8048	8400, 8401, 8411 8421, 8441, 8461
ROM capacity (bytes)	1K	1K	1K, 2K, 4K, 6K ROMless
RAM capacity (bytes)	64	64	64, 64, 128, 128
paralel I/O lines	8 + 8 + 4	8 + 8 + 8	8 + 8 + 4
single inputs	1	3	2
serial I/O	no	no	yes, 2-line multi-transmitter
timer	8 bit	8 bit	8 bit
prescaler	mod. 32	mod. 32	mod. 1 & mod. 32
machine cycle time (μs)	10	2,5	5
for clock (MHz)	3	6	6
instruction set	8021	8048	8048 with omissions; 5 new serial I/O instructions; 2 new register instructions; 2 new control instructions; 1 new cond. branch instruction
interrupts	none	2 external timer/ event counter	3 external serial I/O timer/event counter
no. of pins (DIL)	28	40	28

OSCILLATOR CIRCUITRY

Clock frequency is determined by using the internal oscillator or by connecting an external clock to XTAL1. Where the internal oscillator is used, the frequency is set by a crystal between XTAL1 and XTAL2, or by a ceramic resonator or an inductor, each with two associated capacitors, between XTAL1 and XTAL2 (see Fig. 10). A machine cycle consists of 10 states, each state being 3 oscillator periods. The common 6 MHz crystal gives a 5 μs machine cycle. The MAB84XX family has dynamic logic, and therefore, for adequate refreshing the oscillator frequency must be at least 1 MHz.



1. Crystal – AT-cut
2. Ceramic resonator
   C1 = C2 = 27 pF
   C1 may be trimmed

Fig. 10(a) Quartz crystal or ceramic resonator mode.

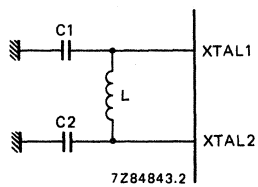
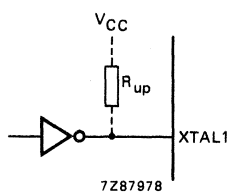


Fig. 10(b) LC pi-network.

LC oscillator timing

frequency	C1 = C2	L
3,0 MHz	33 pF	100 μH
4,0 MHz	33 pF	56 μH
4,4 MHz	33 pF	47 μH
5,0 MHz	33 pF	33 μH
6,0 MHz	33 pF	22 μH



Drive XTAL1
Leave XTAL2 open

Driver may be high-speed CMOS or any TTL
t_r, $t_f < 10$ ns

Fig. 10(c) External drive.

PROGRAM STATUS WORD

The program status word (PSW) is an 8-bit word in the CPU which stores information about the current status of the microcontroller (Fig. 11). The PSW bits are:

bits 0, 1 and 2 – stack pointer bits (SP_0, SP_1, SP_2);

bit 3 – prescaler select (PS); 0 = divide-by-32; 1 = no prescaling;

bit 4 – working register bank select (RBS):
0 = register bank 0
1 = register bank 1;

bit 5 – not used (1);

bit 6 – auxiliary carry (AC):
half-carry bit is generated by an ADD instruction and used by the decimal adjust instruction DA A;

bit 7 – carry (CY):
the carry flag indicates that the previous operation has resulted in an overflow of the accumulator.

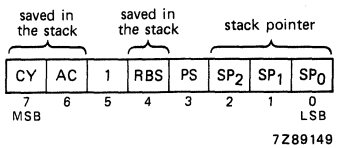


Fig. 11 Program status word.

All bits can be read and written using the MOV A,PSW and MOV PSW,A instructions, respectively.

Bits 6 and 7 can be set and cleared by CPU operation. Bit 4 is changed by the SEL RB instruction, bit 3 by the MOV PSW,A instruction, and bits 0, 1 and 2 by the CALL, RET or RETR instructions and when an interrupt occurs. Bits 4, 6 and 7 are stored in the program counter stack during subroutine and interrupt calls. These bits are restored to the PSW with RETR (return and restore) instruction.

Note: The RET instruction has no restore feature and should not be used at the end of an interrupt because this would leave any further interrupts disabled.

The MAB84XX family has arithmetic, logical and branching capabilities. The DA A, SWAP A, and XCHD instructions simplify BCD arithmetic and the handling of nibbles. The MOVP A,@A instruction permits efficient table look up from the current ROM page.

The conditional branch logic within the processor enables several conditions, internal and external to the processor, to be tested by the user's program. Table 2 lists the conditional branch instructions used to change the program execution sequence. The DJNZ instruction decrements a designated register and branches if the contents are not zero. This instruction makes the register an efficient program loop counter. The JMPP @A instruction allows multiway branches to destinations indirectly addressed by the contents of the accumulator.

Table 2 Conditional branches

TEST	JUMP CONDITION	JUMP INSTRUCTION
accumulator	0 or non-zero	JZ, JNZ
accumulator bit test	1	JB0 to JB7
carry flag	0 or 1	JNC, JC
timer overflow flag	1	JTF
test input $\overline{INT}$	0 or 1	JNI, JI
test input T1	0 or 1	JNT1, JT1
test flag 0	1	JF0
test flag 1	1	JF1
register	non-zero	DJNZ

RESET

A positive-going signal on the RESET input:

- sets the program counter to zero,
- selects location 0 of memory bank 0, and register bank 0,
- sets the stack pointer to zero ('000'B); pointing to RAM address 8,
- disable the interrupts (external, timer and serial I/O),
- stops the timer/event counter, then sets it to zero,
- sets the timer prescaler to divide-by-32,
- resets the timer flag,
- sets all ports to logic '1' (input mode),
- sets the serial I/O to slave receiver mode and disables serial I/O.

The external power-on-reset circuit can consist of a capacitor connected between V_{CC} and the RESET pin. A diode may be added between the RESET pin and ground to ensure reset if the supply voltage falls momentarily. Figure 12(a) and (b) show a typical reset circuit and input characteristics of the RESET pin.

RESET has to be active HIGH for more than 2 machine cycles after the power supply and clock have stabilized.

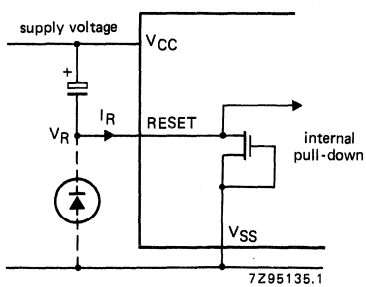


Fig. 12(a) Typical power-on reset circuitry.

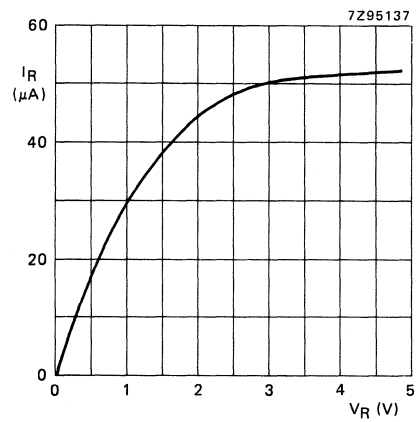


Fig. 12(b) Typical input characteristics (not guaranteed).

INSTRUCTION SET

The instruction set consists of over 80 one and two byte instructions and is based on the MAB8048 instruction set. New instructions include those for serial I/O operation and memory bank selection. Program code efficiency is high because all RAM locations on a 256 byte page require only a single byte address.

Table 3 gives the instruction set of the MAB84XX family and Table 4 shows the instruction map. The following symbols and abbreviations are used.

symbol	description
A	the accumulator
AC	the auxiliary carry flag
addr	program memory address (11-bits)
Bb	bit designation (b = 0–7)
BS	the bank switch
C	carry flag
CLK	clock signal
CNT	event counter
D	nibble designation (4-bits)
DBF	program memory bank flip-flop
data	number or expression (8-bits)
F0, F1	flags 0 and 1
I	interrupt
$\overline{\text{INT}}$	external interrupt
P	'in-page' operation designation
Pp	port designation (p = 1, 2 or 4–7)
PSW	program status word
Rr	register designation (r = 0, 1 or 0–7)
SP	stack pointer
T	timer
TF	timer flag
T0, T1	test 0 and 1 inputs
#	immediate data prefix
@	indirect address prefix
$	current value of program counter
←	is replaced by
↔	is exchanged with

Table 3 MAB84XX family instruction set

	mnemonic	opcode (hex.)	bytes/ cycles	description	function		notes
ACCUMULATOR	ADD A, Rr	6*	1/1	Add register contents to A	(A)←(A) + (Rr)	r = 0–7	1
	ADD A, @Rr	60 61	1/1	Add RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) (A)←(A) + ((R1))		1
	ADD A, #data	03 data	2/2	Add immediate data to A	(A)←(A) + data		1
	ADDC A, Rr	7*	1/1	Add carry and register contents to A	(A)←(A) + (Rr) + (C)	r = 0–7	1
	ADDC A, @Rr	70 71	1/1	Add carry and RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) + (C) (A)←(A) + ((R1)) + (C)		1
	ADDC A, #data	13 data	2/2	Add carry and immediate data to A	(A)←(A) + data + (C)		1
	ANL A, Rr	5*	1/1	'AND' Rr with A	(A)←(A) AND (Rr)	r = 0–7	
	ANL A, @Rr	50 51	1/1	'AND' RAM data, addressed by Rr, with A	(A)←(A) AND ((R0)) (A)←(A) AND ((R1))		
	ANL A, #data	53 data	2/2	'AND' immediate data with A	(A)←(A) AND data		
	ORL A, Rr	4*	1/1	'OR' Rr with A	(A)←(A) OR (Rr)	r = 0–7	
	ORL A, @Rr	40 41	1/1	'OR' RAM data, addressed by Rr, with A	(A)←(A) OR ((R0)) (A)←(A) OR ((R1))		
	ORL A, #data	43 data	2/2	'OR' immediate data with A	(A)←(A) OR data		
	XRL A, Rr	D*	1/1	'XOR' Rr with A	(A)←(A) XOR (Rr)	r = 0–7	
	XRL A, @Rr	D0 D1	1/1	'XOR' RAM, addressed by Rr, with A	(A)←(A) XOR ((R0)) (A)←(A) XOR ((R1))		
	XRL A, #data	D3 data	2/2	'XOR' immediate data with A	(A)←(A) XOR data		
	INC A	17	1/1	increment A by 1	(A)←(A) + 1		
	DEC A	07	1/1	decrement A by 1	(A)←(A) – 1		
	CLR A	27	1/1	clear A to zero	(A)←0		
	CPL A	37	1/1	one's complement A	(A)←NOT(A)		
	RL A	E7	1/1	rotate A left	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(A_7)$	n = 0–6	

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
ACCUMULATOR (cont.)	RLC A	F7	1/1	rotate A left through carry	$(A_{n+1})\leftarrow A_n$ n = 0–6 $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$	2
	RR A	77	1/1	rotate A right	$(A_n)\leftarrow(A_{n+1})$ n = 0–6 $(A_7)\leftarrow(A_0)$	
	RRC A	67	1/1	rotate A right through carry	$(A_n)\leftarrow(A_{n+1})$ n = 0–6 $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$	2
	DA A	57	1/1	decimal adjust A		2
	SWAP A	47	1/1	swap nibbles of A	$(A_{4-7})\leftrightarrow(A_{0-3})$	
DATA MOVES	MOV A, Rr	F*	1/1	move register contents to A	(A)←(Rr) r = 0–7	
	MOV A, @Rr	F0 F1	1/1	move RAM data, addressed by Rr, to A	(A)←((R0)) (A)←((R1))	
	MOV A, #data	23 data	2/2	move immediate data to A	(A)←data	
	MOV Rr, A	A*	1/1	move accumulator contents to register	(Rr)←(A) r = 0–7	
	MOV @Rr, A	A0 A1	1/1	move accumulator contents to RAM location addressed by Rr	((R0))←(A) ((R1))←(A)	
	MOV Rr, #data	B* data	2/2	move immediate data to Rr	(Rr)←data	
	MOV @Rr, #data	B0 data B1 data	2/2	move immediate data to RAM location addressed by Rr	((R0))←data ((R1))←data	
	XCH A, Rr	2*	1/1	exchange accumulator contents with Rr	(A)↔(Rr) r = 0–7	
	XCH A, @Rr	20 21	1/1	exchange accumulator contents with RAM data addressed by Rr	(A)↔((R0)) (A)↔((R1))	
	XCHD A, @Rr	30 31	1/1	exchange lower nibbles of A and RAM data addressed by Rr	$(A_{0-3})\leftrightarrow((R0_{0-3}))$ $(A_{0-3})\leftrightarrow((R1_{0-3}))$	
	MOV A, PSW	C7	1/1	move PSW contents to accumulator	(A)←(PSW)	
	MOV PSW, A	D7	1/1	move accumulator bit 3 to PSW_3	$(PSW_3)\leftarrow(A_3)$	3
	MOVP A, @A	A3	1/2	move indirectly addressed data in current page to A	$(PC_{0-7})\leftarrow(A)$, $(A)\leftarrow((PC))$	

FLAGS	CLR C	97	1/1	clear carry bit	(C)←0		2
	CPL C	A7	1/1	complement carry bit	(C)←NOT(C)		2
REGISTER	INC Rr	1*	1/1	increment register by 1	(Rr)←(Rr) + 1	r = 0–7	
	INC @Rr	10 11	1/1	increment RAM data, addressed by Rr, by 1	((R0))←((R0)) + 1 ((R1))←((R1)) + 1		
	DEC Rr	C*	1/1	decrement register by 1	(Rr)←(Rr) – 1	r = 0–7	
	DEC @Rr	C0 C1	1/1	decrement RAM data, addressed by Rr, by 1	((R0))←((R0)) – 1 ((R1))←((R1)) – 1		
BRANCH	JMP addr	• 4 address	2/2	unconditional jump within a 2K bank	(PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1		
	JMPP @A	B3	1/2	indirect jump within a page	(PC_{0-7})←((A))		
	DJNZ Rr, addr	E* address	2/2	decrement Rr by 1 and jump if not zero to addr	(Rr)←(Rr) – 1 if (Rr) not zero (PC_{0-7})←addr	r = 0–7	
	DJNZ @Rr, addr	E0	2/2	decrement RAM data, addressed by Rr, by 1 and jump if not zero to addr	((R0))←((R0)) – 1 if ((R0)) not zero (PC_{0-7})←addr		
		E1			((R1))←((R1)) – 1 if ((R1)) not zero (PC_{0-7})←addr		
	JBb addr	▲ 2 address	2/2	jump to addr if Acc. bit b = 1	if b = 1: (PC_{0-7})←addr	b = 0–7	
	JC addr	F6 address	2/2	jump to addr if C = 1	if C = 1: (PC_{0-7})←addr		
	JNC addr	E6 address	2/2	jump to addr if C = 0	if C = 0: (PC_{0-7})←addr		
	JZ addr	C6 address	2/2	jump to addr if A = 0	if A = 0: (PC_{0-7})←addr		
	JNZ addr	96 address	2/2	jump to addr if A is NOT zero	if A≠0: (PC_{0-7})←addr		
	JT0 addr	36 address	2/2	jump to addr if T0 = 1	if T0 = 1: (PC_{0-7})←addr		
	JNT0 addr	26 address	2/2	jump to addr if T0 = 0	if T0 = 0: (PC_{0-7})←addr		
	JT1 addr	56 address	2/2	jump to addr if T1 = 1	if T1 = 1: (PC_{0-7})←addr		
	JNT1 addr	46 address	2/2	jump to addr if T1 = 0	if T1 = 0: (PC_{0-7})←addr		
	JTF addr	16 address	2/2	jump to addr if Timer Flag = 1	if TF = 1: (PC_{0-7})←addr		4
	JNTF addr	06 address	2/2	jump to addr if Timer Flag = 0	if TF = 0: (PC_{0-7})←addr		

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
TIMER/EVENT COUNTER	MOV A, T	42	1/1	move timer/event counter contents to accumulator	(A)←(T)	
	MOV T, A	62	1/1	move accumulator contents to timer/event counter	(T)←(A)	
	STRT CNT	45	1/1	start event counter		
	STRT T	55	1/1	start timer		
	STOP TCNT	65	1/1	stop timer/event counter		
	EN TCNTI	25	1/1	enable timer/event counter interrupt		
	DIS TCNTI	35	1/1	disable timer/event counter interrupt		
CONTROL	EN I	05	1/1	enable external interrupt		
	DIS I	15	1/1	disable external interrupt		
	SEL RB0	C5	1/1	select register bank 0	(RBS)←0	5
	SEL RB1	D5	1/1	select register bank 1	(RBS)←1	5
	SEL MB0	E5	1/1	select program memory bank 0	(MBFF0)←0, (MBFF1)←0	
	SEL MB1	F5	1/1	select program memory bank 1	(MBFF0)←1, (MBFF1)←0	
	SEL MB2	A5	1/1	select program memory bank 2	(MBFF0)←0, (MBFF1)←1	
	SEL MB3	B5	1/1	select program memory bank 3	(MBFF0)←1, (MBFF1)←1	
SUBROUTINE	CALL addr	▲ 4 address	2/2	jump to subroutine	((SP))←(PC), ($PSW_{4, 6, 7}$) (SP)←(SP) + 1 (PC_{9-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1	6
	RET	83	1/2	return from subroutine	(SP)←(SP) – 1 (PC)←((SP))	6
	RETR	93	1/2	return from interrupt and restore bits 4, 6, 7 of PSW	(SP)←(SP) – 1 ($PSW_{4, 6, 7}$) + (PC)←((SP))	6

PARALLEL INPUT/OUTPUT	IN A, Pp	08 09 0A	1/2	input port p data to accumulator	(A)←(P0) (A)←(P1) (A)←(P2)	7
	OUTL Pp, A	38 39 3A	1/2	output accumulator data to port p	(P0)←(A) (P1)←(A) (P2)←(A)	
	ANL Pp, #data	98 99 9A	2/2	AND port p data with immediate data	(P0)←(P0) AND data (P1)←(P1) AND data (P2)←(P2) AND data	
	ORL Pp, #data	88 89 8A	2/2	OR port p data with immediate data	(P0)←(P0) OR data (P1)←(P1) OR data (P2)←(P2) OR data	
	OUTL PO,A	90	1/2	Output accumulator data to port ϕ	(P0)←(A)	
SERIAL INPUT/OUTPUT	MOV A, S_n	0C 0D	1/2	move serial I/O register contents to accumulator	(A)←(S0) (A)←(S1)	8
	MOV S_n, A	3C 3D 3E	1/2	move accumulator contents to serial I/O register	(S0)←(A) (S1)←(A) (S2)←(A)	
	MOV S_n, #data	9C 9D 9E	2/2	move immeidate date to serial I/O register	(S0)←data (S1)←data (S2)←data	
	EN SI	85	1/1	enable serial I/O interrupt		
	DIS SI	95	1/1	disable serial I/O interrupt		
	NOP	00	1/1	no operation		

Notes to Table 3.

1. PSW CY, AC affected
2. PSW CY affected
3. PSW PS affected
4. Execution of JTF and JNTF instructions resets the Timer Flag (TF).
5. PSW RBS affected
6. PSW SP_0, SP_1, SP_2 affected
7. (A) = 1111 P23, P22, P21, P20.
8. (S1) has a different meaning for read and write operation, see serial I/O interface.
9. Only for software-transfer from the MAB8021.

* : 8, 9, A, B, C, D, E, F
● : 0, 2, 4, 6, 8, A, C, E
▲ : 1, 3, 5, 7, 9, B, D, F

Table 4 MAB84XX family instruction set

first hexadecimal character of opcode ↓

second hexadecimal character of opcode →

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP			ADD A,#data	JMP page 0	EN I	JNTF addr	DEC A	IN A,Pp				MOV A,Sn			
									0	1	2		0	1		
1	INC @Rr		JBO addr	ADDC A,#data	CALL page 0	DIS I	JTF addr	INC A	INC Rr							
	0	1							0	1	2	3	4	5	6	7
2	XCH A,@Rr			MOV A,#data	JMP page 1	EN TCNTI	JNTO addr	CLR A	XCH A,Rr							
	0	1							0	1	2	3	4	5	6	7
3	XCHD A,@Rr		JB1 addr		CALL page 1	DIS TCNTI	JTO addr	CPL A	OUTL Pp,A				MOV Sn,A			
	0	1							0	1	2		0	1	2	
4	ORL A,@Rr		MOV A,T	ORL A,#data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	ORL A,Rr							
	0	1							0	1	2	3	4	5	6	7
5	ANL A,@Rr		JB2 addr	ANL A,#data	CALL page 2	STRT T	JT1 addr	DA A	ANL A,Rr							
	0	1							0	1	2	3	4	5	6	7
6	ADD A,@Rr		MOV T,A		JMP page 3	STOP TCNT		RRC A	ADD A,Rr							
	0	1							0	1	2	3	4	5	6	7
7	ADDC A,@Rr		JB3 addr		CALL page 3			RR A	ADDC A,Rr							
	0	1							0	1	2	3	4	5	6	7
8				RET	JMP page 4	EN SI			ORL Pp,#data							
									0	1	2					
9	OUTL P0,A		JB4 addr	RETR	CALL page 4	DIS SI	JNZ addr	CLR C	ANL Pp,#data				MOV Sn,#data			
									0	1	2		0	1	2	
A	MOV @Rr,A			MOVP A,@A	JMP page 5	SEL MB2		CPL C	MOV Rr,A							
	0	1							0	1	2	3	4	5	6	7
B	MOV @Rr,#data		JB5 addr	JMPP @A	CALL page 5	SEL MB3			MOV Rr,#data							
	0	1							0	1	2	3	4	5	6	7
C	DEC @Rr				JMP page 6	SEL RB0	JZ addr	MOV A,PSW	DEC Rr							
	0	1							0	1	2	3	4	5	6	7
D	XRL A,@Rr		JB6 addr	XRL A,#data	CALL page 6	SEL RB1		MOV PSW,A	XRL A,Rr							
	0	1							0	1	2	3	4	5	6	7
E	DJNZ @Rr,addr				JMP page 7	SEL MB0	JNC addr	RL A	DJNZ Rr, addr							
	0	1							0	1	2	3	4	5	6	7
F	MOV A,@Rr		JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	MOV A,Rr							
	0	1							0	1	2	3	4	5	6	7

Table 5 shows the additional MAB84XX family instructions (including the five for serial I/O operation) that are not part of the MAB8048 instruction set.

Table 5 MAB84XX family instructions not in the MAB8048 instruction set

serial I/O	register	control	conditional branch

MOV A, S_n
MOV S_n, A
MOV S_n, #data
EN SI
DIS SI
	DEC @Rr
	DJNZ @Rr, addr
		SEL MB2
		SEL MB3
			JNTF addr

Table 6 shows the MAB8048 instructions omitted from the MAB84XX family instruction set.

Table 6 MAB8048 instructions not in the MAB84XX family instruction set

data moves	flags	branch	control

MOVX A, @R
MOVX @R, A
MOVP3 A, @A
MOVD A, P
MOVD P, A
ANLD P, A
ORLD P, A
	CLR F0
	CPL F0
	CLR F1
	CPL F1
		* JNI addr
		JF0 addr
		JF1 addr
		
		* replaced by
		JT0 JNT0.
			ENT0 CLK

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Input voltage on any pin with respect to V_{SS}	V_I		−0,5 to + 7	V
Total power dissipation				
for SOT-117D	P_{tot}	max.	1	W
for SOT-136A (SO-28)	P_{tot}	max.	0,6	W
Input/output current	$\pm I_I$, I_O	max.	10	mA
Storage temperature	T_{stg}		−65 to + 150	°C
Operating temperature				
MAB84XX family (standard)	T_{amb}		0 to + 70	°C
MAF84XX family (extended)	T_{amb}		−40 to + 85	°C
MAF84AXX family (automotive)	T_{amb}		−40 to + 110	°C

D.C. CHARACTERISTICS

V_{CC} = 5 V (± 10%); V_{SS} = 0 V; all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	max.	unit	conditions
Supply current					
MAB	I_{CC}	–	85	mA	0 to + 70 °C
MAF	I_{CC}	–	100	mA	–40 to + 85 °C
MAF84A	I_{CC}	–	100	mA	–40 to + 110 °C
Inputs					
Input voltage LOW (except P23 and SCLK)	V_{IL}	–0,5	0,8	V	
Input voltage LOW (P23 and SCLK)	V_{IL1}	–0,5	1,5	V	
Input voltage HIGH (all inputs except XTAL1, P23 and SCLK)	V_{IH}	2	V_{CC}	V	
Input voltage HIGH (XTAL1, P23 and SCLK)	V_{IH1}	3,0	V_{CC}	V	
Outputs					
Output voltage LOW (P00–P07)	V_{OL}	–	0,45	V	I_{OL} = 1,6 mA
Output voltage LOW (P10–P17)	V_{OL1}	–	0,45	V	I_{OL1} = 1,6 mA
Output voltage LOW (P10, P11 for 8400)	V_{OL11}	–	2,5	V	I_{OL11} = 7 mA ←
Output voltage LOW (P10–P17 for 8401/11/21/41/61)	V_{OL12}	–	1,0	V	I_{OL12} = 10 mA ←
Output voltage LOW (P20–P22)	V_{OL2}	–	0,45	V	I_{OL2} = 1,6 mA
Output voltage LOW (P23, SCLK)	V_{OL3}	–	0,45	V	I_{OL3} = 5 mA
Output voltage LOW (non-standard pins of bond-out versions)	V_{OL4}	–	0,45	V	I_{OL4} = 0,4 mA
Output voltage HIGH (all outputs unless (open drain)	V_{OH}	2,4	–	V	I_{OH} = –50 μA
Output leakage current	± I_{OL}	–	10	μA	$V_{SS} < V_I < V_{CC}$

A.C. CHARACTERISTICS (all versions except bond-out)

V_{CC} = 5 V ± 10%; V_{SS} = 0 V.

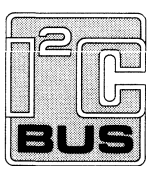
parameter	symbol		min.	max.	unit
Frequency	f_{CL}	MAB/MAF84XX	1	6	MHz
		MAF84AXX	1	5	MHz
Cycle time	t_{CY}	MAB/MAF84XX	5	30	μs
		MAF84AXX	6	30	μs

A.C. CHARACTERISTICS (bond-out versions)

V_{CC} = 5 V ± 10%; V_{SS} = 0 V.

parameter	symbol	min.	max.	unit
f_{CL} = 6 MHz				
Control pulse duration $\overline{PSEN}$ (9CP)	t_{CC}	1,5	9	μs
Address to $\overline{PSEN}$ L set-up (1CP)	t_{AS}	167	–	ns
Data to $\overline{PSEN}$ H set-up (1CP + 120 ns)	t_{DS}	600	–	ns
Data hold time	t_{DR}	0	–	ns
Address to data-in (10CP–t_{DS})	t_{AD}	–	1,07	μs
Time from $\overline{PSEN}$ L to C1 (3CP)	t_{PC}	500	–	ns
Time from $\overline{INTA}$ L to $\overline{PSEN}$ (3CP)	t_{IP0}	500	–	ns
Time from $\overline{INTA}$ H to $\overline{PSEN}$ (6CP)	t_{IP1}	1	–	μs
$\overline{HALT}$ set-up to $\overline{PSEN}$ (15CP)	t_{HS}	2,5	–	μs
$\overline{HALT}$ hold time from $\overline{PSEN}$ (3CP)	t_{HH}	500	–	ns

Note: CP = clock pulse.



Purchase of Philips' I²C components conveys a license under the Philips' I²C patent to use the components in the I²C-system provided the system conforms to the I²C specifications defined by Philips.

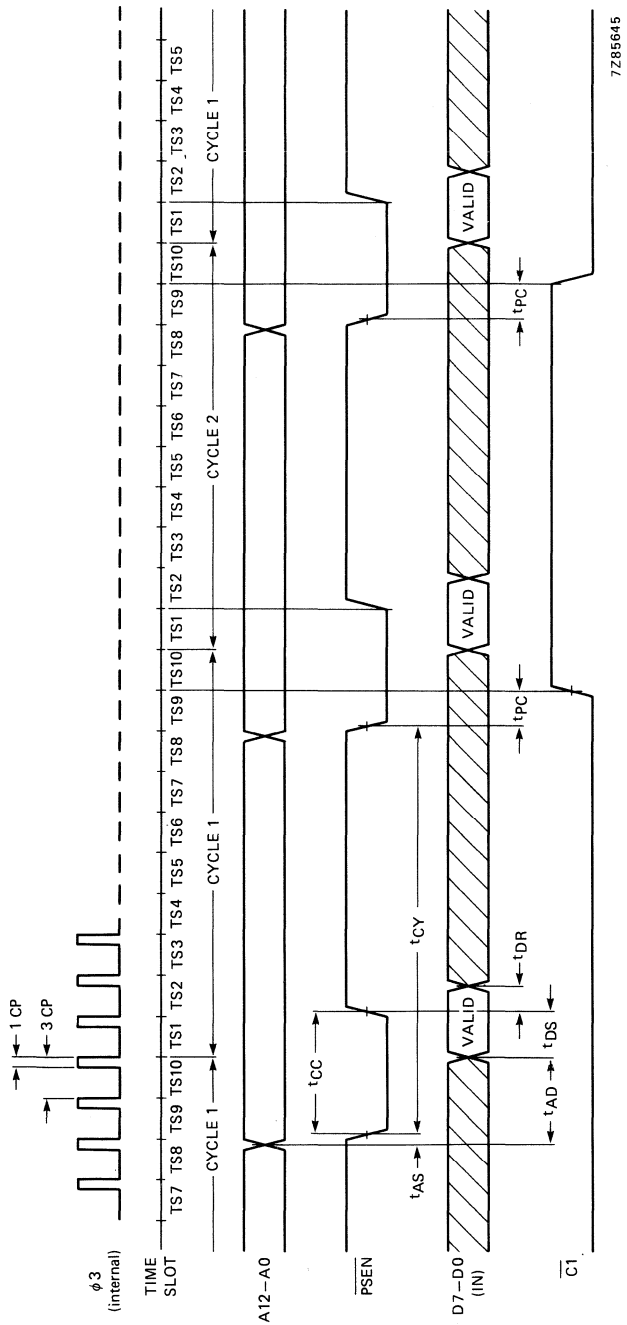


Fig. 13 Memory-access timing MAB8400/01WP/B.

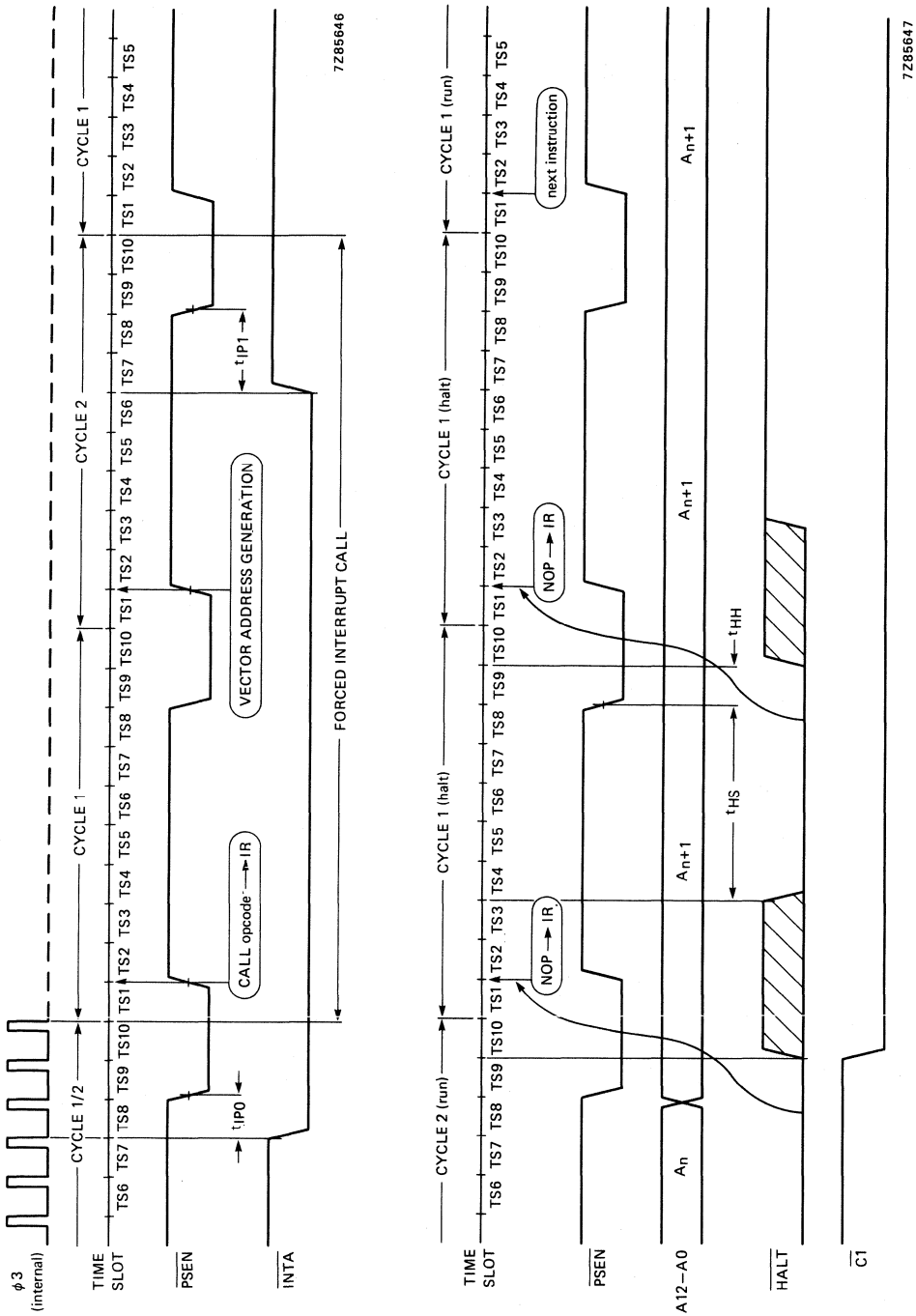


Fig. 14 $\overline{\text{INTA}}$ and $\overline{\text{HALT}}$ timing MAB8400/01WP/B.

SINGLE-CHIP 8-BIT MICROCONTROLLER

The MAB8422/8442 is a high-performance microcontroller incorporating dedicated hardware, memory capacity and I/O lines. This dedication means a microcontroller can be economically installed in high-volume products where its main function is control.

The MAB8422/8442 is a 20 pin, single-chip 8-bit microcontroller that has been developed from the 28 pin MAB8420/8440 microcontrollers. The versions are:

- MAB8422 - 2K ROM/64 RAM bytes plus 8-bit LED-driver
- MAB8442 - 4K ROM/128 RAM bytes plus 8-bit LED-driver

Each version has 15 I/O port lines comprising one 8-bit parallel port (P0), one 2-bit parallel port (P10 and P11 that are shared with the serial I/O lines SDA and SCL), one 3-bit parallel port (P20-P22) and two input lines ($\overline{INT}$/T0 and T1).

The serial I/O interface is I^2C compatible and therefore the MAB8422/8442 can operate as a slave or a master in single and multi-master systems. Conversion from parallel to serial data when transmitting, and vice versa when receiving, is done mainly in software. There is a minimum of hardware for the serial I/O implemented. This hardware is controlled by the status of the SDA and SCL lines and can be read or written under software control. Standard software for I^2C-bus control is available on request.

Features

- 8-bit: CPU, ROM, RAM and I/O
- 20 pin package
- MAB8422: 2K ROM/64 RAM bytes
- MAB8442: 4K ROM/128 RAM bytes
- 13 quasi-bidirectional I/O port lines
- Two testable inputs $\overline{INT}$/T0 and T1
- High current output on P0 (I_{OL} = 10 mA at V_{OL} = 1 V)
- One interrupt line combined with the testable input line $\overline{INT}$/T0
- Single-level interrupts: external, timer/event counter, serial I/O
- I^2C-compatible serial I/O that can be used in single or multi-master systems (serial I/O data and clock via P10 and P11 port lines, respectively)
- 8-bit programmable timer/event counter
- Internal oscillator, generated with inductor, crystal, ceramic resonator or external source
- Over 80 instructions (based on MAB8048)
- All instructions 1 or 2 cycles, cycle time dependent on oscillator frequency
- Single 5 V power supply
- 0 to 70 °C operating temperature range, also versions for −40 to 85 °C and −40 to 110 °C

PACKAGE OUTLINES

MAB/F8422/42P: 20-lead DIL; plastic (SOT-146).
MAF84A22/A42P: 20-lead DIL; plastic (SOT-146).

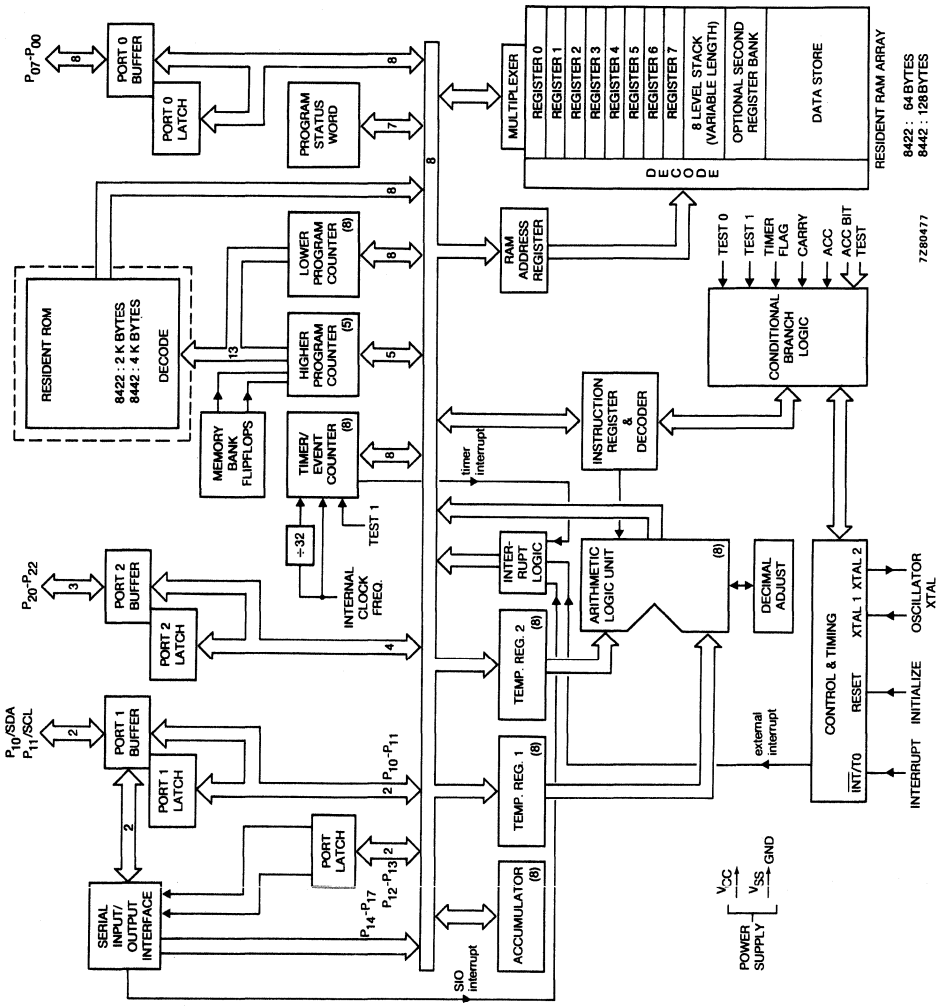


Fig. 1 Block diagram of the MAB8422/8442.

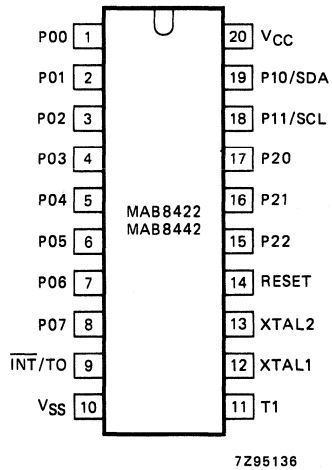


Fig. 2 MAB8422/8442 Pinning diagram.

DEVELOPMENT DATA

PINNING

Designation	Pin number	Function
P00-P07	1-8	8-bit quasi-bidirectional I/O port (Port 0-high current output).
$\overline{INT}$/T0	9	External interrupt input (sensitive to a negative going edge) and/or input, testable using the JT0 or JNT0 instructions.
V_{SS}	10	Ground.
T1	11	Input pin, testable using the JT1 or JNT1 instructions. It can be designated as event counter input using the STRT CNT-instruction. It can also be used to detect zero cross-over of slowly moving a.c. inputs.
XTAL1	12	Connection to timing component that determines the frequency of the internal oscillator. It is also the input for an external clock source.
XTAL2	13	Connection to the other side of the timing component.
RESET	14	Input to initialize the processor (active HIGH).
P10/SDA	19	Quasi-bidirectional port in parallel port mode. Serial data I/O in serial I/O mode.
P11/SCL	18	Quasi-bidirectional port in parallel port mode. Serial clock in serial I/O mode.
P20-P22	17-15	Quasi-bidirectional port.
V_{CC}	20	Power supply.

FUNCTIONAL DESCRIPTION

Program and data memory

The non-volatile program memory (ROM), as shown in Fig. 3, is arranged in two banks of 2K bytes, that are selected by SEL MB instructions, and each bank is further divided into 256-byte pages. Only the unconditional jump instructions (JMP and CALL) can be used to cross page boundaries. Memory bank boundaries can also be crossed using these instructions provided that the appropriate memory bank has been selected.

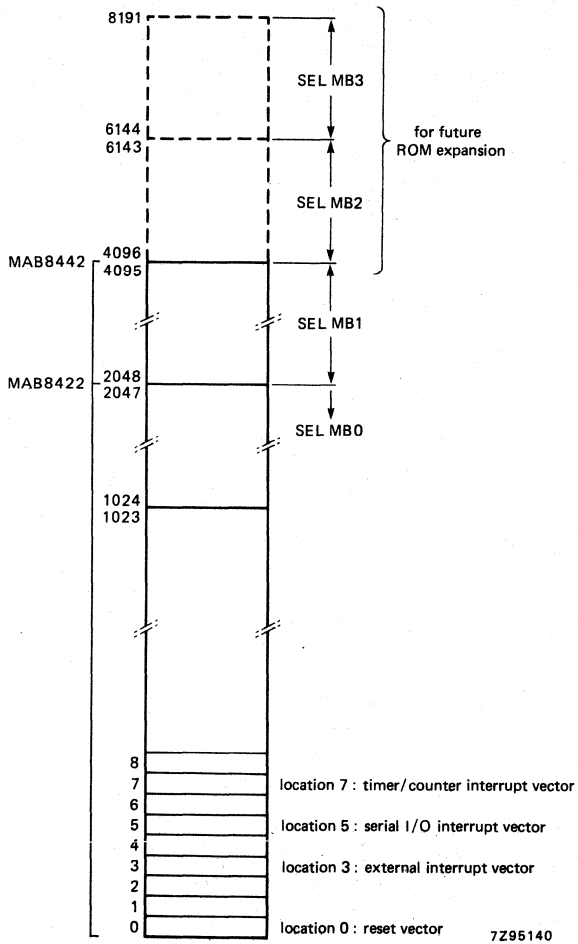


Fig. 3 The program memory map.

In the volatile data memory (RAM), all locations are indirectly addressable using RAM pointer registers and up to 16 designated location can be addressed directly. The memory also includes an 8-level program counter stack addressed by a 3-bit stack pointer and two register banks, each with 8 registers. Fig. 4 illustrates the data memory.

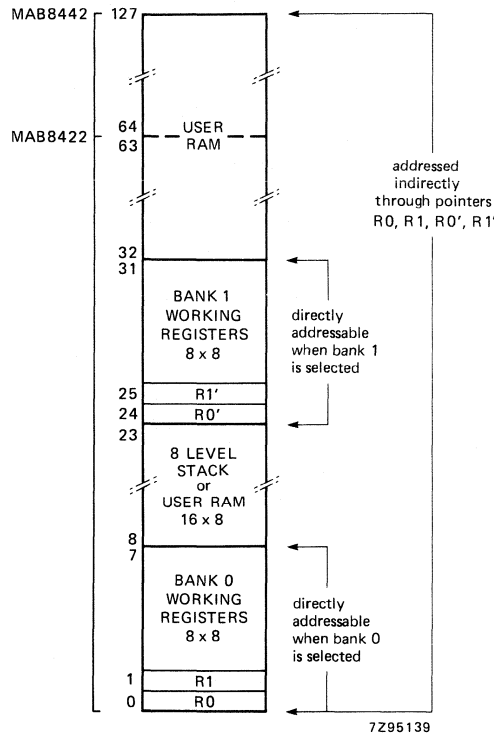


Fig. 4 The data memory map.

Instruction set

The instruction set consists of over 80 one and two byte instructions. It is identical to the MAB8400 instruction set except that the instructions MOV Sn,A, MOV A,Sn and MOV Sn,#data are not used. Program code efficiency is high because all ROM locations on a 256 byte page require only a single byte address.

On-chip peripheral functions

In addition to the CPU and memories, an interrupt system, I/O facilities, and an 8-bit timer/event counter are integrated on-chip to assist the CPU in repetitious, complicated or time-critical tasks. The I/O facilities include the I/O pins, parallel ports and a serial I/O port sharing two pins of a parallel port.

FUNCTIONAL DESCRIPTION (continued)

I/O facilities

The MAB8422/8442 has 13 I/O lines and 2 testable inputs arranged as:

- An 8 line parallel port P00-P07, high current outputs with three optional output configurations: push-pull output with pull-up, open-drain with pull-up and open drain without pull-up
- A 2 line parallel/serial port P10/SDA and P11/SCL, open-drain without pull-up output configuration only, Schmitt-trigger input;
- A 3 line parallel port P20-P22 with output configurations as P00-P07;
- An external interrupt and test input $\overline{\text{INT}}$/T0, which when used as a test input can be tested by the conditional jump instructions JT0 or JNT0;
- A test input T1, tested by the conditional jump instructions JT1 or JNT1. T1 can also be used as an input to the timer/event counter or to detect zero cross-over of slowly moving a.c. signals.

Timer/event counter

An 8-bit binary up-counter is provided. This can count external events, modulo-32 machine cycles, or machine cycles directly. When used as a timer, the input to the counter is either the overflow or input of a 5-bit prescaler. When used as an event counter, LOW to HIGH transitions on T1 (pin 13) are counted. The maximum rate at which the counter may be incremented is once every machine (200 kHz for a 5 μs machine cycle). Fig. 5 illustrates the timer/event counter.

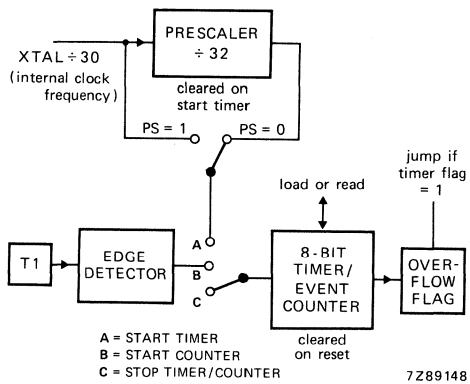


Fig. 5 The timer/event counter.

Interrupt system

External events and real-time on-chip peripherals require servicing by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution a multiple-source, single-level nested interrupt system is provided.

The MAB8422/8442 handles interrupts from three sources as follows:

- $\overline{\text{INT}}$/T0; externally via pin 9
- $\overline{\text{SIOINT}}$; from the internal serial I/O port
- TCNT interrupt; from the internal timer/event counter.

Each interrupt vectors to a separate location in the program memory for its service program. Each source can be individually enabled or disabled. When more than one interrupt occurs simultaneously, their priority will be: (1) external, (2) serial I/O and, (3) timer/event counter. An additional external interrupt can be created using the timer/event counter interrupt.

OSCILLATOR CIRCUITRY

Clock frequency is determined by using the internal oscillator or by connecting an external clock to XTAL1. Where the internal oscillator is used the frequency is set by a crystal between XTAL1 and XTAL2, or by a ceramic resonator or an inductor, each with two associated capacitors, between XTAL1 and XTAL2 (see Fig. 6). A machine cycle consists of 10 states, each state being 3 oscillator periods. The MAB8422/8442 has dynamic logic, and therefore, for adequate refreshing the oscillator frequency must be at least 1 MHz.

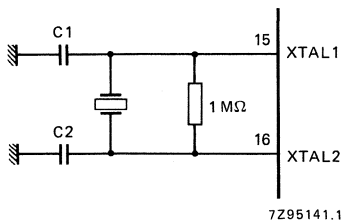


Fig. 6a Quartz crystal or ceramic resonator mode. C1 = C2 = 27 pF

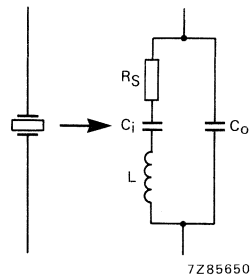


Fig. 6b Quartz crystal equivalent.

DEVELOPMENT DATA

Quartz crystal parameter

Cut, resonance frequency: AT, 1 to 6 MHz (fundamental)

If the frequency has to be trimmed, then a trimmer capacitor should be connected in parallel with the fixed capacitor C_1

Table 1 shows the LC values for timing generation with the LC oscillator.

Table 1 LC oscillator timing

L (μH)	C (pF)	nominal frequency (MHz)
22	33	6,0
33	33	5,0
47	33	4,4
56	33	4,0
100	33	3,0

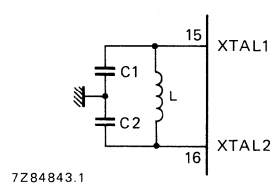


Fig. 7 LC oscillator mode.

RESET

A positive-going signal on the RESET input:

- sets the program counter to zero,
- selects location 0 of memory bank 0, and register bank 0,
- sets the stack pointer to zero (000); pointing to RAM address 8,
- disables the interrupts (external), timer and serial I/O),
- stops the timer/event counter, then sets it to zero,
- sets the timer prescaler to modulo-32,
- resets the timer flag,
- sets all ports to logic '1' (input mode),
- sets ports P10/SDA and P11/SCC to the parallel part mode and disables the serial I/O.

The external power-on-reset circuit can consist of a capacitor connected between V_{CC} and the RESET pin. A diode may be added between the RESET pin and ground to ensure reset if the supply voltage falls momentarily (see Fig. 9a).

RESET has to be active HIGH for a minimum of 2 machine cycles after the power supply and clock have stabilized.

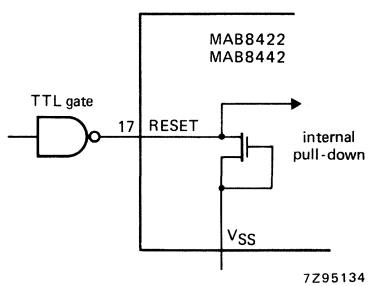


Fig. 8 External reset.

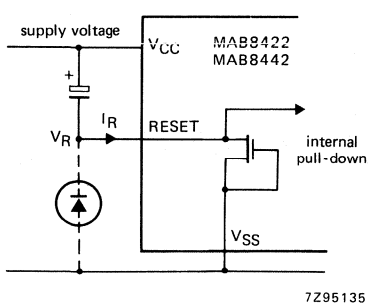


Fig. 9a Typical power-on reset circuitry.

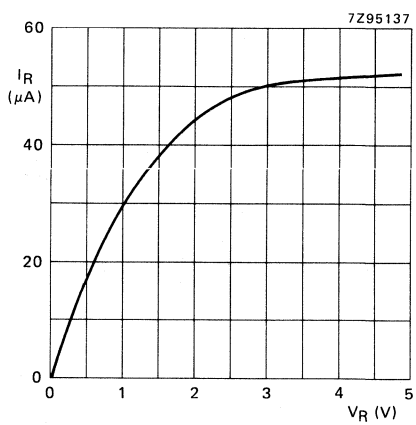


Fig. 9b Typical input characteristics.

INSTRUCTION SET

	mnemonic	opcode (hex.)	bytes/ cycles	description	function		notes
ACCUMULATOR	ADD A, Rr	6*	1/1	Add register contents to A	(A)←(A) + (Rr)	r = 0–7	1
	ADD A, @Rr	60 61	1/1	Add RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) (A)←(A) + ((R1))		1
	ADD A, #data	03 data	2/2	Add immediate data to A	(A)←(A) + data		1
	ADDC A, Rr	7*	1/1	Add carry and register contents to A	(A)←(A) + (Rr) + (C)	r = 0–7	1
	ADDC A, @Rr	70 71	1/1	Add carry and RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) + (C) (A)←(A) + ((R1)) + (C)		1
	ADDC A, #data	13 data	2/2	Add carry and immediate data to A	(A)←(A) + data + (C)		1
	ANL A, Rr	5*	1/1	'AND' Rr with A	(A)←(A) AND (Rr)	r = 0–7	
	ANL A, @Rr	50 51	1/1	'AND' RAM data, addressed by Rr, with A	(A)←(A) AND ((R0)) (A)←(A) AND ((R1))		
	ANL A, #data	53 data	2/2	'AND' immediate data with A	(A)←(A) AND data		
	ORL A, Rr	4*	1/1	'OR' Rr with A	(A)←(A) OR (Rr)	r = 0–7	
	ORL A, @Rr	40 41	1/1	'OR' RAM data, addressed by Rr, with A	(A)←(A) OR ((R0)) (A)←(A) OR ((R1))		
	ORL A, #data	43 data	2/2	'OR' immediate data with A	(A)←(A) OR data		
	XRL A, Rr	D*	1/1	'XOR' Rr with A	(A)←(A) XOR (Rr)	r = 0–7	
	XRL A, @Rr	D0 D1	1/1	'XOR' RAM, addressed by Rr, with A	(A)←(A) XOR ((R0)) (A)←(A) XOR ((R1))		
	XRL A, #data	D3 data	2/2	'XOR' immediate data with A	(A)←(A) XOR data		
	INC A	17	1/1	increment A by 1	(A)←(A) + 1		
	DEC A	07	1/1	decrement A by 1	(A)←(A) −1		
	CLR A	27	1/1	clear A to zero	(A)←0		
	CPL A	37	1/1	one's complement A	(A)←NOT(A)		
	RL A	E7	1/1	rotate A left	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(A_7)$	n = 0–6	

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
ACCUMULATOR (cont.)	RLC A	F7	1/1	rotate A left through carry	$(A_{n+1})\leftarrow A_n$ n = 0–6 $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$	2
	RR A	77	1/1	rotate A right	$(A_n)\leftarrow(A_{n+1})$ n = 0–6 $(A_7)\leftarrow(A_0)$	
	RRC A	67	1/1	rotate A right through carry	$(A_n)\leftarrow(A_{n+1})$ n = 0–6 $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$	2
	DA A	57	1/1	decimal adjust A		2
	SWAP A	47	1/1	swap nibbles of A	$(A_{4-7})\leftrightarrow(A_{0-3})$	
DATA MOVES	MOV A, Rr	F*	1/1	move register contents to A	(A)←(Rr) r = 0–7	
	MOV A, @Rr	F0 F1	1/1	move RAM data, addressed by Rr, to A	(A)←((R0)) (A)←((R1))	
	MOV A, #data	23 data	2/2	move immediate data to A	(A)←data	
	MOV Rr, A	A*	1/1	move accumulator contents to register	(Rr)←(A) r = 0–7	
	MOV @Rr, A	A0 A1	1/1	move accumulator contents to RAM location addressed by Rr	((R0))←(A) ((R1))←(A)	
	MOV Rr, #data	B* data	2/2	move immediate data to Rr	(Rr)←data	
	MOV @Rr, #data	B0 data B1 data	2/2	move immediate data to RAM location addressed by Rr	((R0))←data ((R1))←data	
	XCH A, Rr	2*	1/1	exchange accumulator contents with Rr	(A)↔(Rr) r = 0–7	
	XCH A, @Rr	20 21	1/1	exchange accumulator contents with RAM data addressed by Rr	(A)↔((R0)) (A)↔((R1))	
	XCHD A, @Rr	30 31	1/1	exchange lower nibbles of A and RAM data addressed by Rr	$(A_{0-3})\leftrightarrow((R0_{0-3}))$ $(A_{0-3})\leftrightarrow((R1_{0-3}))$	
	MOV A, PSW	C7	1/1	move PSW contents to accumulator	(A)←(PSW)	
	MOV PSW, A	D7	1/1	move accumulator bit 3 to PSW_3	$(PSW_3)\leftarrow(A_3)$	3
	MOVP A, @A	A3	1/2	move indirectly addressed data in current page to A	$(PC_{0-7})\leftarrow(A)$, $(A)\leftarrow((PC))$	

FLAGS	CLR C	97	1/1	clear carry bit	(C)←0		2
	CPL C	A7	1/1	complement carry bit	(C)←NOT(C)		2
REGISTER	INC Rr	1*	1/1	increment register by 1	(Rr)←(Rr) + 1	r = 0–7	
	INC @Rr	10 11	1/1	increment RAM data, addressed by Rr, by 1	((R0))←((R0)) + 1 ((R1))←((R1)) + 1		
	DEC Rr	C*	1/1	decrement register by 1	(Rr)←(Rr) – 1	r = 0–7	
	DEC @Rr	C0 C1	1/1	decrement RAM data, addressed by Rr, by 1	((R0))←((R0)) – 1 ((R1))←((R1)) – 1		
BRANCH	JMP addr	• 4 address	2/2	unconditional jump within a 2 K bank	(PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1		
	JMPP @A	B3	1/2	indirect jump within a page	(PC_{0-7})←((A))		
	DJNZ Rr, addr	E* address	2/2	decrement Rr by 1 and jump if not zero to addr	(Rr)←(Rr) – 1 if (Rr) not zero (PC_{0-7})←addr	r = 0–7	
	DJNZ @Rr, addr	E0	2/2	decrement RAM data, addressed by Rr by 1 and jump if not zero to addr	((R0))←((R0)) – 1 if ((R0)) not zero (PC_{0-7})←addr		
		E1			((R1))←((R1)) – 1 if ((R1)) not zero (PC_{0-7})←addr		
	JBb addr	▲ 2 address	2/2	jump to addr if Acc. bit b = 1	if b = 1 : (PC_{0-7})←addr	b = 0–7	
	JC addr	F6 address	2/2	jump to addr if C = 1	if C = 1 : (PC_{0-7})←addr		
	JNC addr	E6 address	2/2	jump to addr if C = 0	if C = 0 : (PC_{0-7})←addr		
	JZ addr	C6 address	2/2	jump to addr if A = 0	if A = 0 : (PC_{0-7})←addr		
	JNZ addr	96 address	2/2	jump to addr if A is NOT zero	if A≠0 : (PC_{0-7})←addr		
	JT0 addr	36 address	2/2	jump to addr if T0 = 1	if T0 = 1: (PC_{0-7})←addr		
	JNT0 addr	26 address	2/2	jump to addr if T0 = 0	if T0 = 0: (PC_{0-7})←addr		
	JT1 addr	56 address	2/2	jump to addr if T1 = 1	if T1 = 1: (PC_{0-7})←addr		
	JNT1 addr	46 address	2/2	jump to addr if T1 = 0	if T1 = 0: (PC_{0-7})←addr		
	JTF addr	16 address	2/2	jump to addr if Timer Flag = 1	if TF = 1: (PC_{0-7})←addr		4
	JNTF addr	06 address	2/2	jump to addr if Timer Flag = 0	if TF = 0: (PC_{0-7})←addr		

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
TIMER/EVENT COUNTER	MOV A, T	42	1/1	move timer/event counter contents to accumulator	(A)←(T)	
	MOV T, A	62	1/1	move accumulator contents to timer/event counter	(T)←(A)	
	STRT CNT	45	1/1	start event counter		
	STRT T	55	1/1	start timer		
	STOP TCNT	65	1/1	stop timer/event counter		
	EN TCNTI	25	1/1	enable timer/event counter interrupt		
	DIS TCNTI	35	1/1	disable timer/event counter interrupt		
CONTROL	EN I	05	1/1	enable external interrupt		
	DIS I	15	1/1	disable external interrupt		
	SEL RB0	C5	1/1	select register bank 0	(RBS)←0	5
	SEL RB1	D5	1/1	select register bank 1	(RBS)←1	5
	SEL MB0	E5	1/1	select program memory bank 0	(MBFF0)←0, (MBFF1)←0	
	SEL MB1	F5	1/1	select program memory bank 1	(MBFF0)←1, (MBFF1)←0	
	SEL MB2	A5	1/1	select program memory bank 2	(MBFF0)←0, (MBFF1)←1	
	SEL MB3	B5	1/1	select program memory bank 3	(MBFF0)←1, (MBFF1)←1	
SUBROUTINE	CALL addr	▲ 4 address	2/2	jump to subroutine	((SP))←(PC), ($PSW_{4, 6, 7}$) (SP)←(SP) + 1 (PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1	6
	RET	83	1/2	return from subroutine	(SP)←(SP) – 1 (PC)←((SP))	6
	RETR	93	1/2	return from interrupt and restore bits 4, 6, 7 of PSW	(SP)←(SP) – 1 ($PSW_{4, 6, 7}$) + (PC)←((SP))	6

PARALLEL INPUT/OUTPUT	IN A, Pp	08 09 0A	1/2	input port p data to accumulator	(A)←(P0) (A)←(P1) (A)←(P2)	7
	OUTL Pp, A	38 39 3A	1/2	output accumulator data to port p	(P0)←(A) (P1)←(A) (P2)←(A)	
	ANL Pp, #data	98 99 9A	2/2	AND port p data with immediate data	(P0)←(P0) AND data (P1)←(P1) AND data (P2)←(P2) AND data	
	ORL Pp, #data	88 89 8A	2/2	OR port p data with immediate data	(P0)←(P0) OR data (P1)←(P1) OR data (P2)←(P2) OR data	
SERIAL INPUT/OUTPUT	EN SI	85	1/1	enable serial I/O interrupt		
	DIS SI	95	1/1	disable serial I/O interrupt		
	NOP	00	1/1	no operation		

Notes

1. PSW CY, AC affected
2. PSW CY affected
3. PSW PS affected
4. Execution of JTF and JNTF instructions resets the Timer Flag (TF).
5. PSW RBS affected
6. PSW SP_0, SP_1, SP_2 affected
7. (A) = 1111 P23, P22, P21, P20.

* : 8, 9, A, B, C, D, E, F
● : 0, 2, 4, 6, 8, A, C, E
▲ : 1, 3, 5, 7, 9, B, D, F

TABLE 1. MAB8422/8442 INSTRUCTION MAP

first hexadecimal character of opcode

second hexadecimal character of opcode

▼	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP			ADD A,#data	JMP page 0	EN I	JNTF addr	DEC A	IN A,Pp							
1	INC @Rr 0	1	JBO addr	ADDC A,#data	CALL page 0	DIS I	JTF addr	INC A	INC Rr 0	1	2	3	4	5	6	7
2	XCH A,@Rr 0	1		MOV A,#data	JMP page 1	EN TCNTI	JNTO addr	CLR A	XCH A,Rr 0	1	2	3	4	5	6	7
3	XCHD A,@Rr 0	1	JB1 addr		CALL page 1	DIS TCNTI	JTO addr	CPL A	OUTL Pp,A 0	1	2					
4	ORL A,@Rr 0	1	MOV A,T	ORL A,#data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	ORL A,Rr 0	1	2	3	4	5	6	7
5	ANL A,@Rr 0	1	JB2 addr	ANL A,#data	CALL page 2	STRT T	JT1 addr	DA A	ANL A,Rr 0	1	2	3	4	5	6	7
6	ADD A,@Rr 0	1	MOV T,A		JMP page 3	STOP TCNT		RRC A	ADD A,Rr 0	1	2	3	4	5	6	7
7	ADDC A,@Rr 0	1	JB3 addr		CALL page 3			RR A	ADDC A,Rr 0	1	2	3	4	5	6	7
8				RET	JMP page 4	EN SI			ORL Pp,#data 0	1	2					
9			JB4 addr	RETR	CALL page 4	DIS SI	JNZ addr	CLR C	ANL Pp,#data 0	1	2					
A	MOV @Rr,A 0	1		MOVP A,@A	JMP page 5	SEL MB2		CPL C	MOV Rr,A 0	1	2	3	4	5	6	7
B	MOV @Rr,#data 0	1	JB5 addr	JMPP @A	CALL page 5	SEL MB3			MOV Rr,#data 0	1	2	3	4	5	6	7
C	DEC @Rr 0	1			JMP page 6	SEL RB0	JZ addr	MOV A,PSW	DEC Rr 0	1	2	3	4	5	6	7
D	XRL A,@Rr 0	1	JB6 addr	XRL A,#data	CALL page 6	SEL RB1		MOV PSW,A	XRL A,Rr 0	1	2	3	4	5	6	7
E	DJNZ @Rr,addr 0	1			JMP page 7	SEL MB0	JNC addr	RL A	DJNZ Rr, addr 0	1	2	3	4	5	6	7
F	MOV A,@Rr 0	1	JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	MOV A,Rr 0	1	2	3	4	5	6	7

DEVELOPMENT DATA

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Input voltage on any pin with respect to ground (V_{SS})			V_I	−0,5 to + 7	V
Total power dissipation			P_{tot}	max. 1	W
Input/output current			I_I, I_O	max. 20	mA
Storage temperature			T_{stg}	−65 to + 150	°C
Operating temperature	MAB8422/42	(standard)	T_{amb}	0 to + 70	°C
	MAF8422/42	(extended)	T_{amb}	−40 to + 85	°C
	MAF84A22/42	(automotive)	T_{amb}	−40 to + 110	°C

D.C. CHARACTERISTICS

V_{CC} = 5 V (± 10%); V_{SS} = 0 V

parameter	symbol		min.	max.	unit	conditions
Supply current	I_{CC}	MAB8422/42	–	70	mA	at 0 °C
		MAF8422/42		90		at −40 °C
		MAF84A22/42		90		at −40 °C
Inputs						
Input voltage LOW (except P10/SDA and P11/SCL)	V_{IL}		−0,5	0,8	V	
Input voltage LOW (P10/SDA and P11/SCL)	V_{IL1}		−0,5	1,5	V	
Input voltage HIGH all inputs except XTAL 1, P10/SDA and P11/SCL	V_{IH}		2,0	V_{CC} + 0,5	V	
Input voltage HIGH to XTAL 1, P10/SDA and P11/SCL	V_{IH1}		3,0	V_{CC} + 0,5	V	
Outputs						
Output voltage LOW (P0 only)	V_{OL}			1,0	V	I_{OL} = 10 mA
Output voltage LOW (P10/SDA and P11/SCL) (note 1)	V_{OL1}			0,45	V	I_{OL} = 5 mA
Output voltage LOW (P0 and P2)	V_{OL2}			0,45	V	I_{OL} = 1,6 mA
Output voltage HIGH (all outputs unless open-drain)	V_{OH}		2,4		V	I_{OH} = −50 μA
Output leakage current	I_{OL}			10	μA	$V_{CC} > V_I > V_{SS}$

A.C. CHARACTERISTICS

parameter	symbol	min.	max.	unit	conditions
Frequency	f_{XTAL}	1	6	MHz	
Cycle time	t_{CY}	5	30	μs	6 MHz crystal = 5 μs

T1 ZERO-CROSS CHARACTERISTICS

T_{amb} = 0 to + 70 °C; V_{CC} = 5 V ± 10%; V_{SS} = 0 V; C_L = 80 pF

parameter	symbol	min.	max.	unit	conditions
Zero-cross detection input (T1) peak-to-peak	$V_{ZX(p-p)}$	1	3	V	a.c. coupled, C = 0,2 μF
Zero-cross accuracy	A_{ZX}	–	± 135	mV	50 Hz sine wave
Zero-cross detection input frequency (T1)	F_{ZX}	0,05	1	kHz	

SINGLE-CHIP 8-BIT MICROCONTROLLER

DESCRIPTION

The PCB80C51 family of single-chip 8-bit microcontrollers is manufactured in an advanced CMOS process. The family consists of the following members:

- PCB80C31: ROM-less version of the PCB80C51
- PCB80C51: 4 K bytes mask-programmable ROM, 128 bytes RAM

In the following, the generic term "PCB80C51" is used to refer to both family members.

The device provides hardware features, architectural enhancements and new instructions to function as a controller for applications requiring up to 64 K bytes of program memory and/or up to 64 K bytes of data storage.

The PCB80C51 contains a non-volatile 4 K x 8 read-only program memory (not ROM-less version); a volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/event counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and timing circuits. For systems that require extra capability, the PCB80C51 can be expanded using standard TTL compatible memories and logic.

The PCB80C31/80C51 has two software selectable modes of reduced activity for further power reduction — Idle and Power Down.
The Idle modes freezes the CPU while allowing the RAM, timers, serial port and interrupt system to continue functioning.
The Power Down mode saves the RAM contents but freezes the oscillator causing all other chip functions to be inoperative.

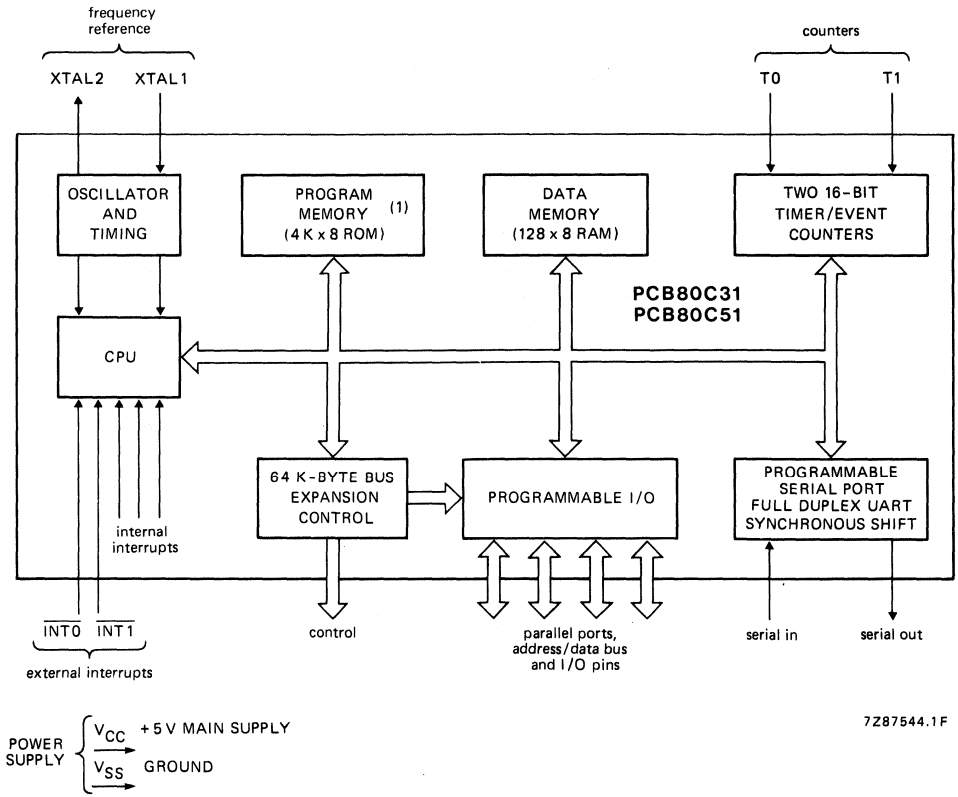
The device also functions as an arithmetic processor having facilities for both binary and BCD arithmetic plus bit-handling capabilities. The instruction set consists of 255 instructions; 44% one-byte, 41% two-byte and 15% three-byte. With a 12 MHz crystal, 58% of the instructions are executed in 1 μs and 40% in 2 μs. Multiply and divide instructions require 4 μs. Multiply, divide, subtract and compare are among the many instructions added to the standard PCB80C48 instruction set.
Software development to be announced: PCB85C51 in piggy-back and 84 pin PLCC.

Features

- 4 K x 8 ROM (80C51 only), 128 x 8 RAM
- Four 8-bit ports, 32 I/O lines
- Two 16-bit timer/event counters
- Full-duplex serial port
- External memory expandable to 128 K, external ROM up to 64 K and/or external RAM up to 64 K
- Boolean processing
- 218 bit-addressable locations
- On-chip oscillator
- Five-source interrupt structure with two priority levels
- 58% of instructions executed in 1 μs; multiply and divide in 4 μs; all others executed in 2 μs (at 12 MHz clock)
- Enhanced architecture with:
  non-page-oriented-instructions
  direct addressing
  four 8-byte + 1 byte register banks
  stack depth up to 128-bytes
  multiply, divide, subtract and compare instructions.

PACKAGE OUTLINES

PCB80C31/51P: 40-lead DIL; plastic (SOT-129).
PCB80C31/51WP: 44-lead PLCC; plastic, leaded-chip-carrier (SOT-187A).



(1) PCB80C51 only.

Fig. 1 Block diagram.

DEVELOPMENT DATA

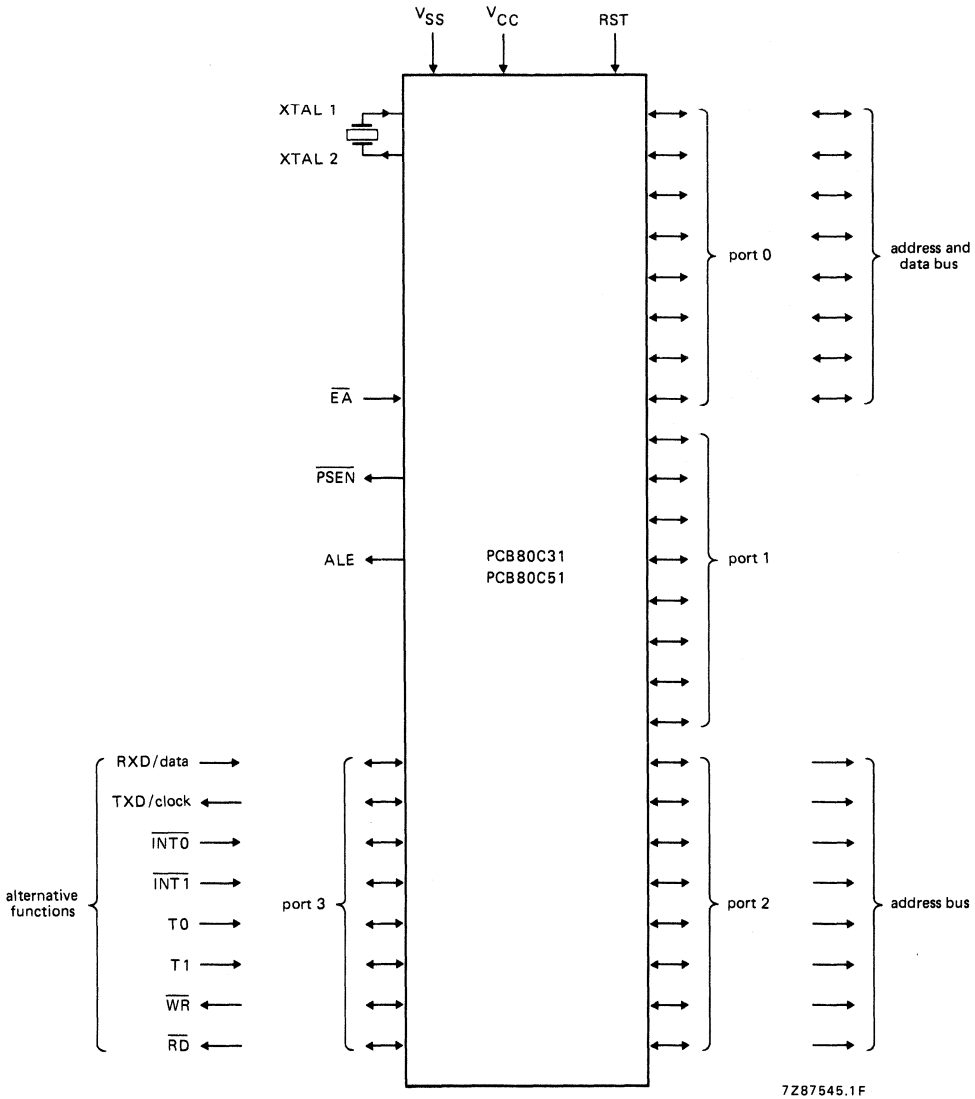


Fig. 2 Functional diagram.

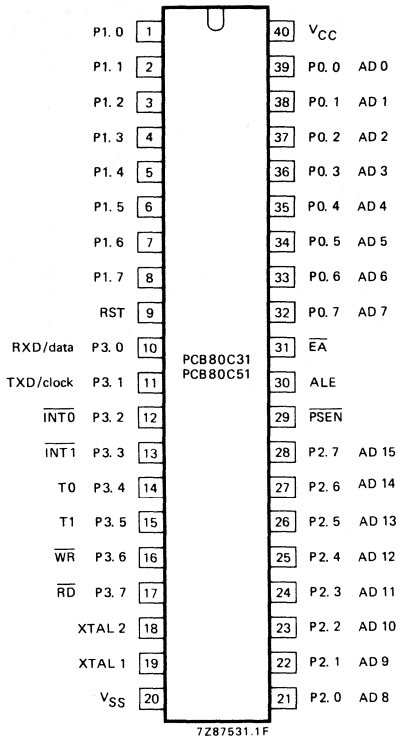


Fig. 3a Pinning diagram for PCB80C31/51P.

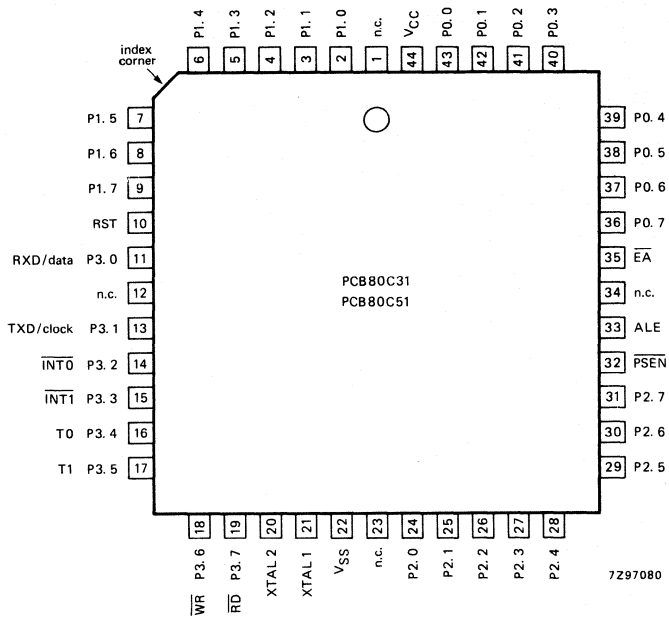


Fig. 3b Pinning diagram for PCB80C31/51WP.

PINNING (PCB80C31/51P)

1–8	P1.0–P1.7	**Port 1**: 8-bit quasi-bidirectional I/O port. Port 1 can sink/source one TTL (= 4 LS TTL) input. It can drive CMOS inputs without external pull-ups.
9		**RST**: a high level on this pin for two machine cycles while the oscillator is running resets the device. An internal pull-down permits Power-On reset using only a capacitor connected to V_{CC}.
10–17	P3.0–P3.7	**Port 3**: 8-bit quasi-bidirectional I/O port with internal pull-ups. It also serves the following alternative functions:

Port pin	*Alternative function*
P3.0	**RXD/data**: serial port receiver data input (asynchronous) or data input/output (synchronous)
P3.1	**TXD/clock**: serial port transmitter data output (asynchronous) or clock output (synchronous)
P3.2	**$\overline{\text{INT0}}$**: external interrupt 0 or gate control input for timer/event counter 0
P3.3	**$\overline{\text{INT1}}$**: external interrupt 1 or gate control input for timer/event counter 1

		P3.4 **T0** : external input for timer/event counter 0
		P3.5 **T1** : external input for timer/event counter 1
		P3.6 **$\overline{WR}$** : external data memory write strobe
		P3.7 **$\overline{RD}$** : external data memory read strobe
		Operation of an alternative function is determined by the relevant output latch programmed to logic 1. Port 3 can sink/source one TTL input. It can drive CMOS inputs without external pull-ups.
18	XTAL 2	**Crystal input 2**: output of the inverting amplifier that forms the oscillator. Left open-circuit when an external oscillator is used (see figures 8 and 9).
19	XTAL 1	**Crystal input 1**: input to the inverting amplifier that forms the oscillator, and input to the internal clock generator. Receives the external oscillator signal when an external oscillator is used (see figures 8 and 9).
20	V_{SS}	**Ground**: circuit ground potential.
21–28	P2.0–P2.7	**Port 2**: 8-bit quasi-bidirectional I/O port with internal pull-ups. Port 2 can sink/source one TTL input. It can drive CMOS inputs without external pull-ups. During the access to external memories (RAM/ROM) that uses 16-bit addresses (MOVX @DPTR) Port 2 emits the high-order address byte. When used for an external RAM an 8-bit address (MOVX @Ri) Port 2 emits the contents of the P2 Special Function Register.
29	$\overline{PSEN}$	**Program Store Enable output**: read strobe to the external Program Memory. It is activated twice each machine cycle during fetches from external Program Memory. When executing out of external Program Memory two activations of $\overline{PSEN}$ are skipped during each access to external Data Memory. $\overline{PSEN}$ is not activated (remains HIGH) during fetches from internal Program Memory. $\overline{PSEN}$ can sink/source 8 LS TTL inputs. It can drive CMOS inputs without an external pull-up.
30	ALE	**Address Latch Enable output**: latches the low byte of the address during accesses to external memory in normal operation. It is activated every six oscillator periods except during an external data memory access. ALE can sink/source 8 LS TTL inputs. It can drive CMOS inputs without an external pull-up.
31	$\overline{EA}$	**External Access input**: When $\overline{EA}$ is held at a TTL high level the CPU executes out of the internal Program Memory (ROM), provided the Program Counter is less than 4096. When $\overline{EA}$ is held at a TTL low level, the CPU executes out of external Program Memory. $\overline{EA}$ is not allowed to float.
32–39	P0.7–P0.0	**Port 0**: 8-bit open drain bidirectional I/O port. It is also the multiplexed low-order address and data bus during accesses to external memory (during these accesses it activates internal pull-ups). Port 0 can sink/source eight TTL inputs.
40	V_{CC}	**Power supply**: + 5 V power supply pin during normal operation, Idle mode and Power Down mode.

FUNCTIONAL DESCRIPTION

General

The PCB80C51 is a stand-alone high-performance microcontroller designed for use in real-time applications such as instrumentation, industrial control and intelligent computer peripherals.

The device provides hardware features, architectural enhancements and new instructions to function as a controller for applications requiring up to 64 K bytes of program memory and/or up to 64 K bytes of data storage.

The PCB80C31 is a control-oriented CPU without on-chip program memory. It can address 64 K bytes of external program memory in addition to 64 K bytes of external data memory. The PCB80C51 is a PCB80C31 with the lower 4 K bytes of program memory filled with on-chip mask programmable ROM. For systems requiring extra capability, the PCB80C51 can be expanded using standard memories and peripherals.

The two pin-compatible versions of this component reduce development problems to a minimum and provide maximum flexibility. The PCB80C51 is for low-cost, high volume production and the PCB80C31 for applications requiring the flexibility of external program memory which can be easily modified and updated in the field.

The PCB80C51 contains a non-volatile 4 K x 8 read-only program memory; a volatile 128 x 8 read/write data memory; 32 I/O lines; two 16-bit timer/event counters; a five-source, two-priority-level, nested interrupt structure; a serial I/O port for either multi-processor communications, I/O expansion, or full duplex UART; and on-chip oscillator and timing circuits.

Central processing unit

The central processing unit (CPU) manipulates operands in four memory spaces. These are the 64 K-byte external data memory, 384-byte internal data memory, the 64 K-byte internal and external program memory and 16-bit program counter spaces. The internal data memory address space is sub-divided into the 256-byte internal data RAM and 128-byte special function register (SFR) address spaces, as shown in Fig. 4.

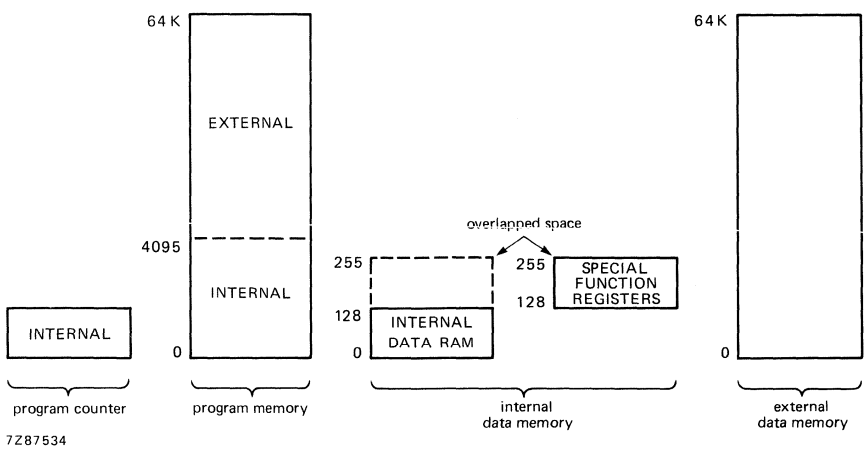


Fig. 4 Memory map.

The internal data RAM contains four register banks (each with eight registers), 128 addressable bits, and the stack. The stack depth is limited by the available internal data RAM and its location is determined by the 8-bit stack pointer. All registers except the program counter and the four 8-register banks reside in the special function register address space. These memory mapped registers include arithmetic registers, pointers, I/O ports, interrupt system registers, timers and serial port. There are 128 addressable bite locations in the SFR address space.

The PCB80C51 contains 128 bytes of internal data RAM and 20 special function registers. It provides a non-paged program memory address space to accommodate relocatable code. Conditional branches are performed relative to the program counter. The register-indirect jump permits branching relative to a 16-bit base register with an offset provided by an 8-bit index register. 16-bit jumps and calls permit branching to any location in the contiguous 64 K program memory address space.

The PCB80C51 has five methods for addressing source operands:

- Register.
- Direct.
- Register-Indirect.
- Immediate.
- Base-Register-plus Index-Register-Indirect.

The first three methods can be used for addressing destination operands. Most instructions have a "destination/source" field that specifies the data type, addressing methods and operands involved. For operations other than moves, the destination operand is also a source operand.

Access addressing is as follows:

- Registers in the four 8-register banks through Register, Direct, or Register-Indirect.
- 128 bytes of internal data RAM through Direct or Register-Indirect.
- Special function registers through Direct.
- External data memory through Register-Indirect.
- Program memory look-up tables through Base-Register-plus Index-Register-Indirect.

The PCB80C51 is classified as an 8-bit device since the internal ROM, RAM, Special Function Registers (SFR), Arithmetic Logic Unit (ALU), and external data bus are each 8-bits wide. It performs operations on bit, nibble, byte and double-byte data types.

Facilities are available for byte transfer, logic, and integer arithmetic operations. Data transfer, logic, and conditional branch operations can be performed directly on Boolean variables to provide excellent bit handling.

DEVELOPMENT DATA

I/O facilities

The PCB80C51 has 32 I/O lines treated as 32 individual addressable bits and as four parallel 8-bit addressable ports. Ports 0, 1, 2 and 3 perform the following alternate functions:

- Port 0; provides the multiplexed low-order address and data bus used for expanding the PCB80C51 with standard memories and peripherals.
- Port 2; provides the high-order address bus when expanding the PCB80C51 with external program memory or more than 256 bytes of external data memory.
- Port 3; pins can be configured individually to provide:

  external interrupt requests inputs
  counter inputs
  serial port receiver input and transmitter output
  control signals to READ and WRITE to external data memory

The generation or use of a Port 3 pin as an alternate function is carried out automatically by the PCB80C51 provided the pin is loaded with a HIGH content.

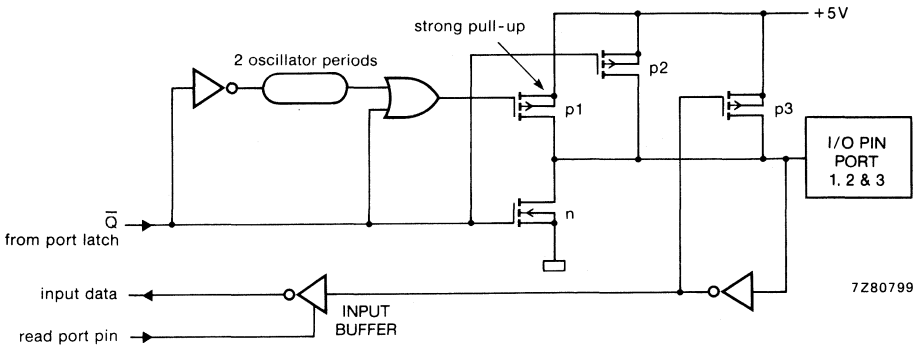


Fig. 5 I/O buffers in the PCB80C51 (Ports 1, 2 and 3).

Timer/event counters

The PCB80C51 contains two 16-bit registers, Timer 0 and Timer 1, that can be used as timers or event counters to carry out the following functions:

- Measure time intervals and pulse durations.
- Count events.
- Generate interrupt requests.

Each timer/event counter can be programmed independently to operate in three modes:

- Mode 0; 8-bit timer or 8-bit counter each with divide by 32 prescaler.
- Mode 1; 16-bit time-interval or event counter.
- Mode 2; 8-bit time-interval or event counter with automatic reload upon overflow.

Counter 0 can be programmed to operate in an additional mode as follows:

- Mode 3; one 8-bit time-interval or event counter and one 8-bit time-interval counter.

When counter 0 is in Mode 3, counter 1 can be programmed to operate in Modes 0, 1 or 2 but cannot set an interrupt request flag or generate an interrupt. However the overflow from counter 1 can be used to pulse the serial Port transmission-rate generator.

The frequency handling range of these counters with a 12 MHz crystal is as follows:

- Up to 1 MHz when programmed for an input that is a division by 12 of the oscillator frequency.
- 0 Hz to an upper limit of 150 kHz to 0,5 MHz when programmed for external inputs.

Both internal and external inputs can be gated to the counter by a second external source for directly measuring pulse durations.

The counters are started and stopped under software control. Each one sets its interrupt request flag when it overflows from all 1's to all 0's (or automatic reload value).

On-chip peripheral functions

In addition to the CPU and memories, an interrupt system, extensive I/O facilities, and several peripheral functions are integrated on-chip to relieve the CPU of repetitious, complicated or time-critical tasks and to permit stringent real-time control of external system interfaces. The I/O facilities include the I/O pins, parallel ports, bidirectional address/data bus and the serial port for I/O expansion. The CPU peripheral functions integrated on-chip are the two 16-bit timer/event counters and the serial port.

Idle and Power-down operation (see Fig. 6)

The Power-down operation freezed the oscillator. The Idle mode operation allows the interrupt, serial port and timer blocks to continue to function while the clock to the CPU is halted.

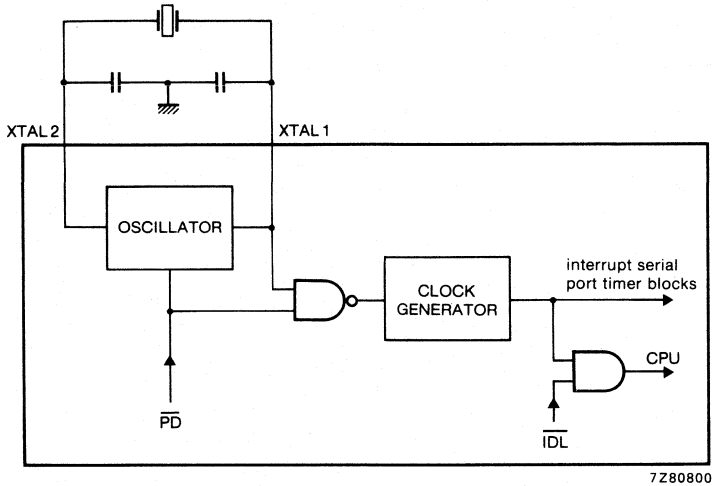


Fig. 6 Internal Idle and Power-down clock configuration.

Power control register (PCON)

These special modes are activated by software via the Special Function Register PCON. Its hardware address is 87H. PCON is not bit addressable.

MSB							LSB
SMOD	–	–	–	GF1	GF0	PD	IDL

symbol	position	name and function
SMOD	PCON.7	Double Baud rate bit when set to logic 1 the baud rate is doubled when the serial port is being used in either modes 1, 2 or 3
–	PCON.6	(reserved)
–	PCON.5	(reserved)
–	PCON.4	(reserved)
GF1	PCON.3	general-purpose flag bit
GF0	PCON.2	general-purpose flag bit
PD	PCON.1	Power-down bit setting this bit activates power-down operation
IDL	PCON.0	Idle mode bit setting this bit activates the idle mode operation

If logic Is are written to PD and IDL at the same time, PD takes precedence. The reset value of PCON is (0XXX0000).

DEVELOPMENT DATA

Idle mode

The instruction that sets PCON.0 is the last instruction executed in the normal operating mode before Idle mode is activated. Once in the Idle mode, the CPU status is preserved in its entirety: the Stack Pointer, Program Counter, Program Status Word, Accumulator, RAM and all other registers maintain their data during Idle mode. The status of the external pins during Idle mode is shown in Table 1.

There are two ways to terminate the Idle mode:—

Activation of any enabled interrupt will cause PCON.0 to be cleared by hardware terminating Idle mode. The interrupt is serviced, and following return from interrupt instruction RET1, the next instruction to be executed will be the one which follows the instruction that wrote a logic 1 to PCON.0. The flag bits GF0 and GF1 may be used to determine whether the interrupt was received during normal execution or during the Idle mode. For example, the instruction that writes the PCON.0 can also set or clear one or both flag bits. When Idle mode is terminated by an interrupt, the service routine can examine the status of the flags bits.

The second way of terminating the Idle mode is with a hardware reset. Since the oscillator is still running, the hardware reset is required to be active for two machine cycles (24 oscillator periods) to complete the reset operation.

Power-down mode

The instruction that sets PCON.1 is the last executed prior to going into the Power-down mode. Once in Power-down mode, the oscillator is stopped. Only the contents of the on-chip RAM are preserved. The Special Function Registers are not saved. A hardware reset is the only way of exiting the Power-down mode.

In the Power-down mode, V_{CC} may be reduced to minimize circuit power consumption. The voltage must not be reduced until the Power-down mode is entered, but must be restored before the hardware reset is applied which will free the oscillator. Reset should not be released until the oscillator has restarted and stabilized.

The status of the external pins during Power-down mode is shown in Table 1. If the Power-down mode is activated while in external program memory, the port data that is held in the Special Function Register P2 is restored to Port 2. If the data is a logic 1, the port pin is held HIGH during the Power-down mode by the strong pull-up transistor pl (see Fig. 5).

Table 1 Status of the external pins during Idle and Power-down modes.

mode	memory	ALE	$\overline{PSEN}$	Port 0	Port 1	Port 2	Port 3
Idle	internal	1	1	port data	port data	port data	port data
Idle	external	1	1	floating	port data	address	port data
Power-down	internal	0	0	port data	port data	port data	port data
Power-down	external	0	0	floating	port data	port data	port data

DEVELOPMENT DATA

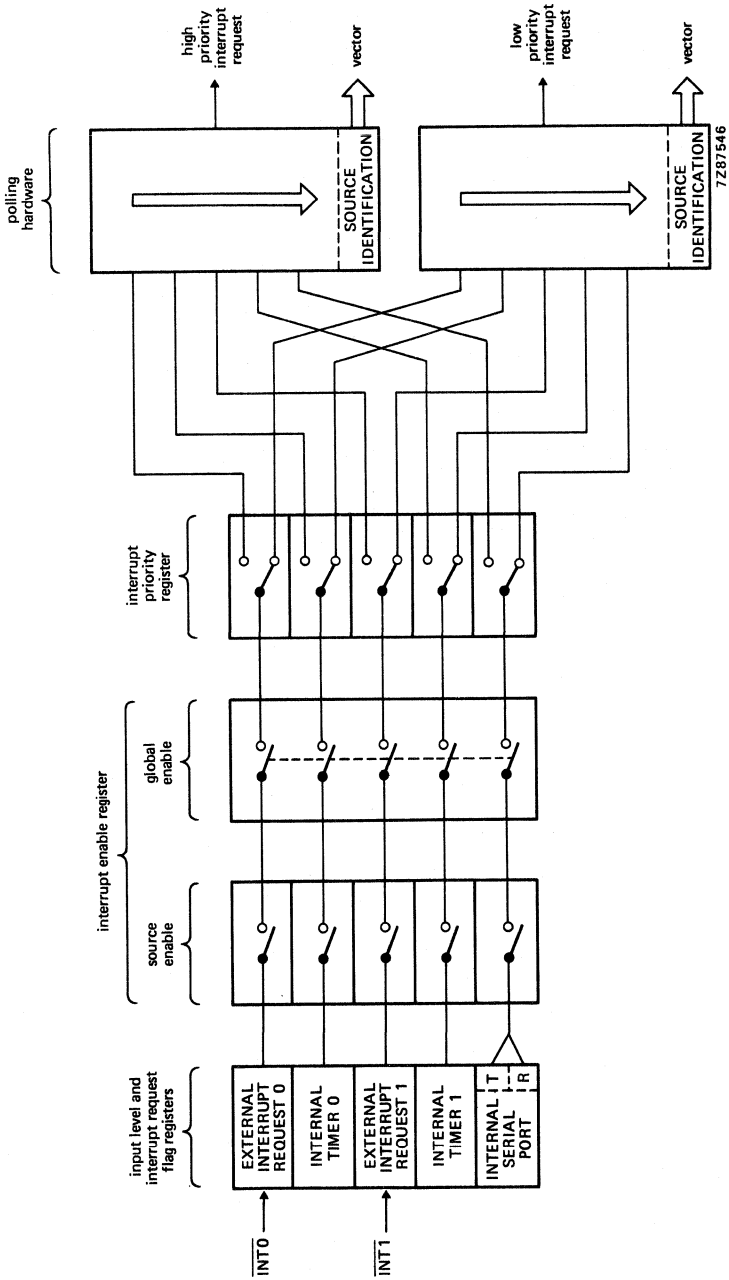


Fig. 7 Interrupt system.

Interrupt system (see Fig. 7)

External events and the real-time-driven on-chip peripherals require service by the CPU asynchronous to the execution of any particular section of code. To tie the asynchronous activities of these functions to normal program execution a multiple-source, two-priority-level, nested interrupt system is provided. Interrupt response latency is from 3 μs to 7 μs when using a 12 MHz crystal.

The PCB80C51 acknowledges interrupt requests from five sources as follows:

- $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$; externally via pins 12 and 13 respectively.
- Timer 0 and Timer 1; from the two internal counters.
- Serial Port; from the internal serial I/O port.

Each interrupt vectors to a separate location in program memory for its service program.

Each source can be individually enabled or disabled and can be programmed to a high or low priority level. Also all enabled sources can be globally disabled or enabled. Both external interrupts can be programmed to be level-activated or transition-activated and is active LOW to allow "wire-ORing" of several interrupt sources to the input pin.

Oscillator circuitry

The oscillator circuitry of the PCB80C51 is a single-stage inverting amplifier in a Pierce oscillator configuration. The circuitry between XTAL 1 and XTAL 2 is basically an inverter biased to the transfer point. Either a crystal or ceramic resonator can be used as the feedback element to complete the oscillator circuitry. Both are operated in parallel resonance. XTAL 1 (pin 19) is the high gain amplifier input, and XTAL 2 (pin 18) is the output (see Fig. 8).

To drive the PCB80C51 externally, XTAL 1 is driven from an external source and XTAL 2 left open-circuit (see Fig. 9).

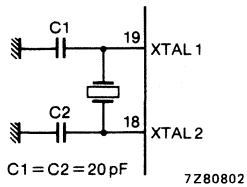


Fig. 8 PCB80C51 oscillator circuit.

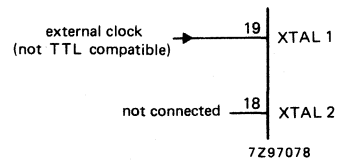


Fig. 9 Driving the PCB80C51 from an external source.

Reset circuitry

The reset circuitry for the PCB80C51 is connected to the reset pin, RST, as shown in Fig. 10. A Schmitt trigger is used at the input for noise rejection. The output of the Schmitt trigger is sampled by the reset circuitry every machine cycle.

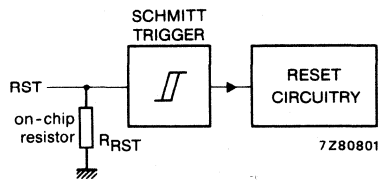


Fig. 10 Reset configuration at RST.

A reset is accomplished by holding the RST pin HIGH for at least two machine cycles (24 oscillator periods), while the oscillator is running. The CPU responds by executing an internal reset. It also configures the ALE and $\overline{\text{PSEN}}$ pins as inputs. (They are quasi-bidirectional.)

The internal reset is executed during the second cycle in which RST is HIGH and is repeated every cycle until RST goes LOW. It leaves the internal registers as follows:

Register	Content
PC	000H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	0000H
P0 - P3	0FFH
IP	(XX000000)
IE	(0X000000)
TMOD	00H
TCON	00H
TH0	00H
TL0	00H
TH1	00H
TL1	00H
SCON	00H
SBUF	Intermediate
PCON	(0XXX0000)

The internal RAM is not affected by reset. When V_{CC} is turned on, the RAM content is indeterminate.

Power-on reset (see Fig. 11)

When V_{CC} is turned on, and provided its rise-time does not exceed 10 ms, an automatic reset can be obtained by connecting the RST pin to V_{CC} via a 10 μF capacitor. When the power is switched on, the current drawn by RST is the difference between V_{CC} and the capacitor voltage, and decreases from V_{CC} as the capacitor charges through the internal resistor (R_{RST}) to ground. The larger the capacitor, the more slowly V_{RST} decreases. V_{RST} must remain above the lower threshold of the Schmitt trigger long enough to effect a complete reset. The time required is the oscillator start-up time, plus 2 machine cycles.

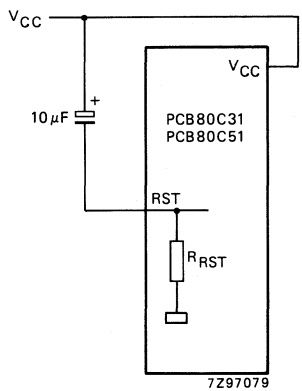


Fig. 11 Power-on reset.

INSTRUCTION SET

The PCB80C51 uses a powerful instruction set to allow expansion of on-chip CPU peripherals and to optimize byte efficiency and execution speed. Reassigned opcodes add new high-power operations and permit new addressing modes to make old operations more orthogonal when compared to the 8048 family.

The instruction set consists of 49 single-byte, 45 two-byte and 17 three-byte instructions. When using a 12 MHz oscillator, 64 instructions execute in 1 μs and 45 instructions execute in 2 μs. Multiply and divide instructions execute in 4 μs.

Table 2 Instruction set description.

mnemonic		description	bytes/ cycles	opcode (hex.)
Arithmetic operation				
ADD	A,Rr	Add register to A	1 1	2*
ADD	A,direct	Add direct byte to A	2 1	25
ADD	A,@Ri	Add indirect RAM to A	1 1	26, 27
ADD	A,#data	Add immediate data to A	2 1	24
ADDC	A,Rr	Add register to A with carry flag	1 1	3*
ADDC	A,direct	Add direct byte to A with carry flag	2 1	35
ADDC	A,@Ri	Add indirect RAM to A with carry flag	1 1	36, 37
ADDC	A,#data	Add immediate data to A with carry flag	2 1	34
SUBB	A,Rr	Subtract register from A with borrow	1 1	9*
SUBB	A,direct	Subtract direct byte from A with borrow	2 1	95
SUBB	A,@Ri	Subtract indirect RAM from A with borrow	1 1	96, 97
SUBB	A,#data	Subtract immediate data from A with borrow	2 1	94
INC	A	Increment A	1 1	04
INC	Rr	Increment register	1 1	0*
INC	direct	Increment direct byte	2 1	05
INC	@Ri	Increment indirect RAM	1 1	06, 07
DEC	A	Decrement A	1 1	14
DEC	Rr	Decrement register	1 1	1*
DEC	direct	Decrement direct byte	2 1	15
DEC	@Ri	Decrement indirect RAM	1 1	16, 17
INC	DPTR	Increment data pointer	1 2	A3
MUL	AB	Multiply A & B	1 4	A4
DIV	AB	Divide A by B	1 4	84
DA	A	Decimal adjust A	1 1	D4

mnemonic		description	bytes/ cycles	opcode (hex.)
Logic operations				
ANL	A,Rr	AND register to A	1 1	5*
ANL	A,direct	AND direct byte to A	2 1	55
ANL	A,@Ri	AND indirect RAM to A	1 1	56, 57
ANL	A,#data	AND immediate data to A	2 1	54
ANL	direct,A	AND A to direct byte	2 1	52
ANL	direct,#data	AND immediate data to direct byte	3 2	53
ORL	A,Rr	OR register to A	1 1	4*
ORL	A,direct	OR direct byte to A	2 1	45
ORL	A,@Ri	OR indirect RAM to A	1 1	46, 47
ORL	A,#data	OR immediate data to A	2 1	44
ORL	direct,A	OR A to direct byte	2 1	42
ORL	direct,#data	OR immediate data to direct byte	3 2	43
XRL	A,Rr	Exclusive-OR register to A	1 1	6*
XRL	A,direct	Exclusive-OR direct byte to A	2 1	65
XRL	A,@Ri	Exclusive-OR indirect RAM to A	1 1	66, 67
XRL	A,#data	Exclusive-OR immediate data to A	2 1	64
XRL	direct, A	Exclusive-OR to direct byte	2 1	62
XRL	direct,#data	Exclusive-OR immediate data to direct byte	3 2	63
CLR	A	Clear A	1 1	E4
CPL	A	Complement A	1 1	F4
RL	A	Rotate A left	1 1	23
RLC	A	Rotate A left through the carry flag	1 1	33
RR	A	Rotate A right	1 1	03
RRC	A	Rotate A right through the carry flag	1 1	13
SWAP	A	Swap nibbles within A	1 1	C4

INSTRUCTION SET (continued)

mnemonic		description	bytes/ cycles	opcode (hex.)
Data transfer				
MOV	A,Rr	Move register to A	1 1	E*
MOV	A,direct	Move direct byte to A	2 1	E5
MOV	A,@Ri	Move indirect RAM to A	1 1	E6, E7
MOV	A,#data	Move immediate data to A	2 1	74
MOV	Rr,A	Move A to register	1 1	F*
MOV	Rr,direct	Move direct byte to register	2 2	A*
MOV	Rr,#data	Move immediate data to register	2 1	7*
MOV	direct,A	Move A to direct byte	2 1	F5
MOV	direct,Rr	Move register to direct byte	2 2	8*
MOV	direct,direct	Move direct byte to direct	3 2	85
MOV	direct,@Ri	Move indirect RAM to direct byte	2 2	86, 87
MOV	direct,#data	Move immediate data to direct byte	3 2	75
MOV	@Ri,A	Move A to indirect RAM	1 1	F6, F7
MOV	@Ri,direct	Move direct byte to indirect RAM	2 2	A6, A7
MOV	@Ri,#data	Move immediate data to indirect RAM	2 1	76, 77
MOV	DPTR,#data16	Load data pointer with a 16-bit constant	3 2	90
MOVC	A,@A+DPTR	Move code byte relative to DPTR to A	1 2	93
MOVC	A,@A+PC	Move code byte relative to PC to A	1 2	83
MOVX	A,@Ri	Move external RAM (8-bit address) to A	1 2	E2, E3
MOVX	A,@DPTR	Move external RAM (16-bit address) to A	1 2	E0
MOVX	@Ri,A	Move A to external RAM (8-bit address)	1 2	F2, F3
MOVX	@DPTR,A	Move A to external RAM (16-bit address)	1 2	F0
PUSH	direct	Push direct byte onto stack	2 2	C0
POP	direct	Pop direct byte from stack	2 2	D0
XCH	A,Rr	Exchange register with A	1 1	C*
XCH	A,direct	Exchange direct byte with A	2 1	C5
XCH	A,@Ri	Exchange indirect RAM with A	1 1	C6, C7
XCHD	A,@Ri	Exchange LOW-order digit indirect RAM with A	1 1	D6, D7

mnemonic		description	bytes/ cycles	opcode (hex.)
Boolean variable manipulation				
CLR	C	Clear carry flag	1 1	C3
CLR	bit	Clear direct bit	2 1	C2
SETB	C	Set carry flag	1 1	D3
SETB	bit	Set direct bit	2 1	D2
CPL	C	Complement carry flag	1 1	B3
CPL	bit	Complement direct bit	2 1	B2
ANL	C,bit	AND direct bit to carry flag	2 2	82
ANL	C,/bit	AND complement of direct bit to carry flag	2 2	B0
ORL	C,bit	OR direct bit to carry flag	2 2	72
ORL	C,/bit	OR complement of direct bit to carry flag	2 2	A0
MOV	C,bit	Move direct bit to carry flag	2 1	A2
MOV	bit,C	Move carry flag to direct bit	2 2	92
Program and machine control				
ACALL	addr11	Absolute subroutine call	2 2	●1addr
LCALL	addr16	Long subroutine call	3 2	12
RET		Return from subroutine	1 2	22
RET1		Return from interrupt	1 2	32
AJMP	addr11	Absolute jump	2 2	▲1addr
LJMP	addr16	Long jump	3 2	02
SJMP	rel	Short jump (relative address)	2 2	80
JMP	@A+DPTR	Jump indirect relative to the DPTR	1 2	73
JZ	rel	Jump if A is zero	2 2	60
JNZ	rel	Jump if A is not zero	2 2	70
JC	rel	Jump if carry flag is set	2 2	40
JNC	rel	Jump if no carry flag	2 2	50
JB	bit,rel	Jump if direct bit is set	3 2	20
JNB	bit,rel	Jump if direct bit is not set	3 2	30
JBC	bit,rel	Jump if direct bit is set and clear bit	3 2	10
CJNE	A,direct,rel	Compare direct to A and jump if not equal	3 2	B5
CJNE	A,#data,rel	Compare immediate to A and jump if not equal	3 2	B4
CJNE	Rr,#data,rel	Compare immed. to reg. and jump if not equal	3 2	B*
CJNE	@Ri,#data,rel	Compare immed. to ind. and jump if not equal	3 2	B6, B7
DJNZ	Rr,rel	Decrement register and jump if not zero	2 2	D*
DJNZ	direct,rel	Decrement direct and jump if not zero	3 2	D5
NOP		No operation	1 1	00

Notes to Table 2

Data addressing modes

Rr	Working register R0-R7.
direct	128 internal RAM locations and any special function register (SFR).
@Ri	Indirect internal RAM location addressed by register R0 or R1 of the actual register bank.
#data	8-bit constant included in instruction.
#data16	16-bit constant included as bytes 2 and 3 of instruction.
bit	direct addressed bit in internal RAM or SFR.
addr16	16-bit destination address. Used by LCALL and LJMP. The branch will be anywhere within the 64 K-byte program memory address space.
addr11	11-bit destination address. Used by ACALL and AJMP. The branch will be within the same 2 K-byte page of program memory as the first byte of the following instruction.
rel	Signed (two's complement) 8-bit offset byte. Used by SJMP and all conditional jumps. Range is −128 to + 127 bytes relative to first byte of the following instruction.

Hexadecimal opcode cross-reference to Table 3

★ : 8, 9, A, B, C, D, E, F.

● : 11, 31, 51, 71, 91, B1, D1, F1.

▲ : 01, 21, 41, 61, 81, A1, C1, E1.

Table 3 Instruction map

first hexadecimal chacter of opcode ↓ — second hexadecimal character of opcode →

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	AJMP addr11	LJMP addr16	RR A	INC A	INC dir	INC @Ri 0	1	INC Rr 0	1	2	3	4	5	6	7
1	JBC bit,rel	ACALL addr11	LCALL addr16	RRC A	DEC A	DEC dir	DEC @Ri 0	1	DEC Rr 0	1	2	3	4	5	6	7
2	JB bit,rel	AJMP addr11	RET	RL A	ADD A,#data	ADD A,dir	ADD A,@Ri 0	1	ADD A,Rr 0	1	2	3	4	5	6	7
3	JNB bit,rel	ACALL addr11	RET1	RLC A	ADDC A,#data	ADDC A,dir	ADDC A,@Ri 0	1	ADDC A,Rr 0	1	2	3	4	5	6	7
4	JC rel	AJMP addr11	ORL dir,A	ORL dir,#data	ORL A,#data	ORL A,dir	ORL A,@Ri 0	1	ORL A,Rr 0	1	2	3	4	5	6	7
5	JNC rel	ACALL addr11	ANL dir,A	ANL dir,#data	ANL A,#data	ANL A,dir	ANL A,@Ri 0	1	ANL A,Rr 0	1	2	3	4	5	6	7
6	JZ rel	AJMP addr11	XRL dir,A	XRL dir,#data	XRL A,#data	XRL A,dir	XRL A,@Ri 0	1	XRL A,Rr 0	1	2	3	4	5	6	7
7	JNZ rel	ACALL addr11	ORL C,bit	JMP @A+DPTR	MOV A,#data	MOV dir,#data	MOV @Ri,#data 0	1	MOV Rr,#data 0	1	2	3	4	5	6	7
8	SJMP rel	AJMP addr11	ANL C,bit	MOVC A,@A+PC	DIV AB	MOV dir,dir	MOV dir,@Ri 0	1	MOV dir,Rr 0	1	2	3	4	5	6	7
9	MOV DPTR, #data	ACALL addr11	MOV bit,C	MOVC A,@A+DPTR	SUBB A,#data	SUBB A,dir	SUBB A,@Ri 0	1	SUBB A,Rr 0	1	2	3	4	5	6	7
A	ORL C,/bit	AJMP addr11	MOV C,bit	INC DPTR	MUL AB		MOV @Ri,dir 0	1	MOV Rr,dir 0	1	2	3	4	5	6	7
B	ANL C,/bit	ACALL addr11	CPL bit	CPL C	CJNE A, #data,rel	CJNE A,dir,rel	CJNE @Ri,#data,rel 0	1	CJNE Rr,#data,rel 0	1	2	3	4	5	6	7
C	PUSH dir	AJMP addr11	CLR bit	CLR C	SWAP A	XCH A,dir	XCH A,@Ri 0	1	XCH A,Rr 0	1	2	3	4	5	6	7
D	POP dir	ACALL addr11	SETB bit	SETB C	DA A	DJNZ dir,rel	XCHD A,@Ri 0	1	DJNZ Rr,rel 0	1	2	3	4	5	6	7
E	MOVX A,@DPTR	AJMP addr11	MOVX A,@Ri 0	1	CLR A	MOV A,dir	MOV A,@Ri 0	1	MOV A,Rr 0	1	2	3	4	5	6	7
F	MOVX @DPTR,A	ACALL addr11	MOVX @Ri,A 0	1	CPL A	MOV dir,A	MOV @Ri,A 0	1	MOV Rr,A 0	1	2	3	4	5	6	7

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Input voltage on any pin with respect to ground (V_{SS})	V_I	−0,5 to + 7	V
Input, output current	$\pm I_I$, I_O	max. 10	mA
Total power dissipation	P_{tot}	max. 1	W
Storage temperature range	T_{stg}	−65 to + 150	°C
Operating ambient temperature range	T_{amb}	0 to + 70	°C

D.C. CHARACTERISTICS

V_{CC} = 5 V (±10%); V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; all voltages with respect to V_{SS} unless otherwise specified

parameter	symbol	min.	max.	unit	conditions
Supply voltage	V_{CC}	4,5	5,5	V	
Supply current					
operating (note 1)	I_{CC}	–	30*	mA	f_{CLK} = 12 MHz
idle mode (note 2)	I_{CC}	–	10*	mA	f_{CLK} = 12 MHz
Power-down current	I_{PD}	–	100*	µA	V_{CC} = 2 V (note 3)
Inputs					
LOW level input voltage (except $\overline{EA}$)	V_{IL}	−0,5	$0{,}2V_{CC}$ −0,1	V	
LOW level input voltage ($\overline{EA}$)	V_{IL1}	−0,5	$0{,}2V_{CC}$ −0,3	V	
HIGH level input voltage (except XTAL 1, RST)	V_{IH}	$0{,}2V_{CC}$ +0,9	V_{CC} +0,5	V	
HIGH level input voltage (XTAL 1, RST)	V_{IH1}	$0{,}7V_{CC}$	V_{CC} +0,5	V	
Input current logic 0 (Ports 1, 2 and 3)	$-I_{IL}$	–	50	µA	V_I = 0,45 V
Input current logic 1 to 0 transition (Ports 1, 2 and 3)	$-I_{TL}$	–	500	µA	V_I = 2 V
Input leakage current (Port 0, $\overline{EA}$)	$\pm I_{LI}$	–	10	µA	0,45 V < V_I < V_{CC}

* Preliminary value.

parameter	symbol	min.	max.	unit	conditions
Outputs					
LOW level output voltage (note 4) (Ports 1, 2 and 3)	V_{OL}	–	0,45	V	I_{OL} = 1,6 mA
LOW level output voltage (note 4) (Port 0, ALE, $\overline{PSEN}$)	V_{OL1}	–	0,45	V	I_{OL} = 3,2 mA
HIGH level output voltage (Ports 1, 2 and 3)	V_{OH}	2,4	–	V	$-I_{OH}$ = 80 μA; V_{CC} = 5 V ± 10%
		$0{,}75V_{CC}$	–	V	$-I_{OH}$ = 30 μA
		$0{,}9V_{CC}$	–	V	$-I_{OH}$ = 10 μA
HIGH level output voltage (note 5) (Port 0 in external Bus mode, ALE, $\overline{PSEN}$)	V_{OH1}	2,4	–	V	$-I_{OH}$ = 400 μA; V_{CC} = 5 V ± 10%
		$0{,}75V_{CC}$	–	V	$-I_{OH}$ = 150 μA
		$0{,}9V_{CC}$	–	V	$-I_{OH}$ = 40 μA
RST pull-down resistor	R_{RST}	40	125	kΩ	
I/O pin capacitance	$C_{I/O}$	–	10	pF	test freq. = 1 MHz; T_{amb} = 25 °C

DEVELOPMENT DATA

Notes to the d.c. characteristics

1. The operating supply current is measured with all output pins disconnected; XTAL 1 driven with $t_r = t_f$ = 10 ns, $V_{IL} = V_{SS}$ + 0,5 V, $V_{IH} = V_{CC}$ –0,5 V; XTAL 2 not connected; $\overline{EA}$ = RST = Port 0 = V_{CC}.
2. The idle mode supply current is measured with all output pins disconnected; XTAL 1 driven with $t_r = t_f$ = 10 ns, $V_{IL} = V_{SS}$ + 0,5 V, $V_{IH} = V_{CC}$ –0,5 V; XTAL 2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
3. The power-down current is measured with all output pins disconnected; XTAL 2 not connected; $\overline{EA}$ = Port 0 = V_{CC}; RST = V_{SS}.
4. Capacitive loading on Port 0 and Port 2 may cause spurious noise pulses to be superimposed on the LOW level output voltage of ALE, Port 1 and Port 3. The noise is due to external Bus capacitance discharging into the Port 0 and Port 2 pins when these pins make a 1-to-0 transition during Bus operations. In the most adverse condition (capacitive loading > 100 pF) the noise pulse on ALE line may exceed 0,8 V. In this event it may be required to qualify ALE with a Schmitt trigger, or use an address latch with a Schmitt trigger STROBE input.
5. Capacitive loading on Port 0 and Port 2 may cause the HIGH level output voltage on ALE and $\overline{PSEN}$ to momentarily fall below the $0{,}9V_{CC}$ specification when the address bits are stabilizing.

A.C. CHARACTERISTICS

V_{CC} = 5 V ± 10%; V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; C_L = 100 pF (Port 0, ALE and $\overline{PSEN}$); C_L = 80 pF (all other outputs); unless otherwise specified (see waveforms Figs 14, 15 and 16)

parameter	symbol	10 MHz		12 MHz		variable clock		unit
		min.	max.	min.	max.	min.	max.	
Program memory								
ALE pulse duration	t_{LL}	160	–	127	–	$2t_{CK}-40$	–	ns
Address set-up time to ALE	t_{AL}	60	–	43	–	$t_{CK}-40$	–	ns
Address hold time after ALE	t_{LA}	65	–	48	–	$t_{CK}-35$	–	ns
Time from ALE to valid instruction input	t_{LIV}	–	300	–	233	–	$4t_{CK}-100$	ns
Time from ALE to control pulse $\overline{PSEN}$	t_{LC}	75	–	58	–	$t_{CK}-25$	–	ns
Control pulse duration $\overline{PSEN}$	t_{CC}	265	–	215	–	$3t_{CK}-35$	–	ns
Time from $\overline{PSEN}$ to valid instruction input	t_{CIV}	–	175	–	125	–	$3t_{CK}-125$	ns
Input instruction hold time after $\overline{PSEN}$	t_{CI}	0	–	0	–	0	–	ns
Input instruction float delay after $\overline{PSEN}$*	t_{CIF}	–	80	–	63	–	$t_{CK}-20$	ns
Address valid after $\overline{PSEN}$*	t_{AC}	92	–	75	–	$t_{CK}-8$	–	ns
Address to valid instruction input	t_{AIV}	–	385	–	302	–	$5t_{CK}-115$	ns
Address float time to $\overline{PSEN}$	t_{AFC}	–12	–	–12	–	0	–	ns

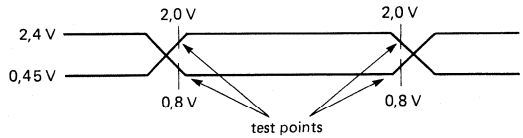
* Interfacing the PCB80C51 to devices with float times up to 75 ns is permitted. This limited bus contention will not cause damage to port 0 drivers.

parameter	symbol	10 MHz		12 MHz		variable clock		unit
		min.	max.	min.	max.	min.	max.	
External data memory								
$\overline{RD}$ pulse duration	t_{RR}	500	–	400	–	$6t_{CK}-100$	–	ns
$\overline{WR}$ pulse duration	t_{WW}	500	–	400	–	$6t_{CK}-100$	–	ns
Address hold time after ALE	t_{LA}	65	–	48	–	$t_{CK}-35$	–	ns
$\overline{RD}$ to valid data input	t_{RD}	–	335	–	250	–	$5t_{CK}-165$	ns
Data hold time after $\overline{RD}$	t_{DR}	0	–	0	–	0	–	ns
Data float delay after $\overline{RD}$	t_{DFR}	–	130	–	97	–	$2t_{CK}-70$	ns
Time from ALE to valid data input	t_{LD}	–	650	–	517	–	$8t_{CK}-150$	ns
Address to valid data input	t_{AD}	–	735	–	585	–	$9t_{CK}-165$	ns
Time from ALE to $\overline{RD}$ or $\overline{WR}$	t_{LW}	250	350	200	300	$3t_{CK}-50$	$3t_{CK}+50$	ns
Time from address to $\overline{RD}$ or $\overline{WR}$	t_{AW}	270	–	203	–	$4t_{CK}-130$	–	ns
Time from $\overline{RD}$ or $\overline{WR}$ HIGH to ALE HIGH	t_{WHLH}	60	140	43	123	$t_{CK}-40$	$t_{CK}+40$	ns
Data valid to $\overline{WR}$ transition	t_{DWX}	40	–	23	–	$t_{CK}-60$	–	ns
Data set-up time before $\overline{WR}$	t_{DW}	550	–	433	–	$7t_{CK}-150$	–	ns
Data hold time after $\overline{WR}$	t_{WD}	50	–	33	–	$t_{CK}-50$	–	ns
Address float delay after $\overline{RD}$	t_{AFR}	–	12	–	12	–	12	ns

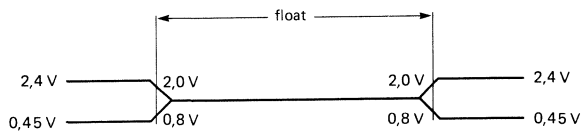
Where:

$1/t_{CK}$ = 3,5 to 12 MHz (see Fig. 13 and Table 4).

A.C. CHARACTERISTICS (continued)



(a)



(b)

A.C. testing inputs are driven at 2,4 V for a logic 1 and 0,45 V for a logic 0.
Timing measurements are taken at 2,0 V for a logic 1 and 0,8 V for logic 0.
The float state is defined as the point at which a Port 0 pin sinks 3,2 mA or sources 400 μA at the voltage test levels.

Fig. 12 A.C. testing input, output waveform (a) and float waveform (b).

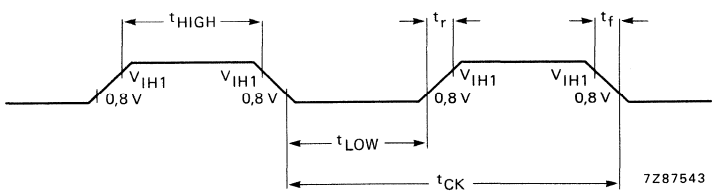


Fig. 13 External clock drive XTAL 1 (see Table 4).

Table 4 External clock drive XTAL 1 (see Fig. 13)

parameter	symbol	variable clock (f = 3,5 to 12 MHz) min.	variable clock (f = 3,5 to 12 MHz) max.	unit
oscillator clock period	t_{CK}	83,3	286	ns
HIGH time	t_{HIGH}	20	$t_{CK}-t_{LOW}$	ns
LOW time	t_{LOW}	20	$t_{CK}-t_{HIGH}$	ns
rise time	t_r	–	20	ns
fall time	t_f	–	20	ns

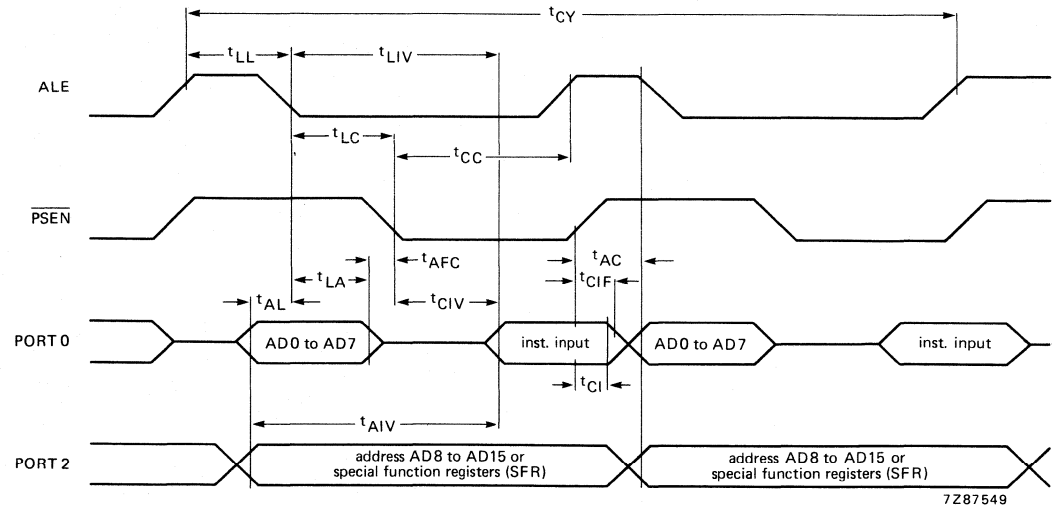


Fig. 14 Read from program memory.

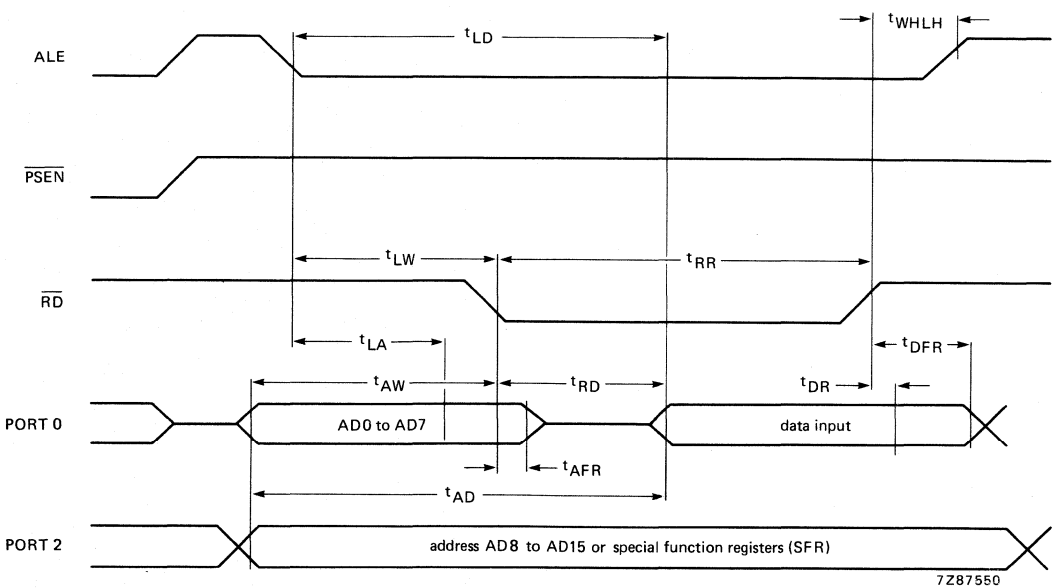


Fig. 15 Read from data memory.

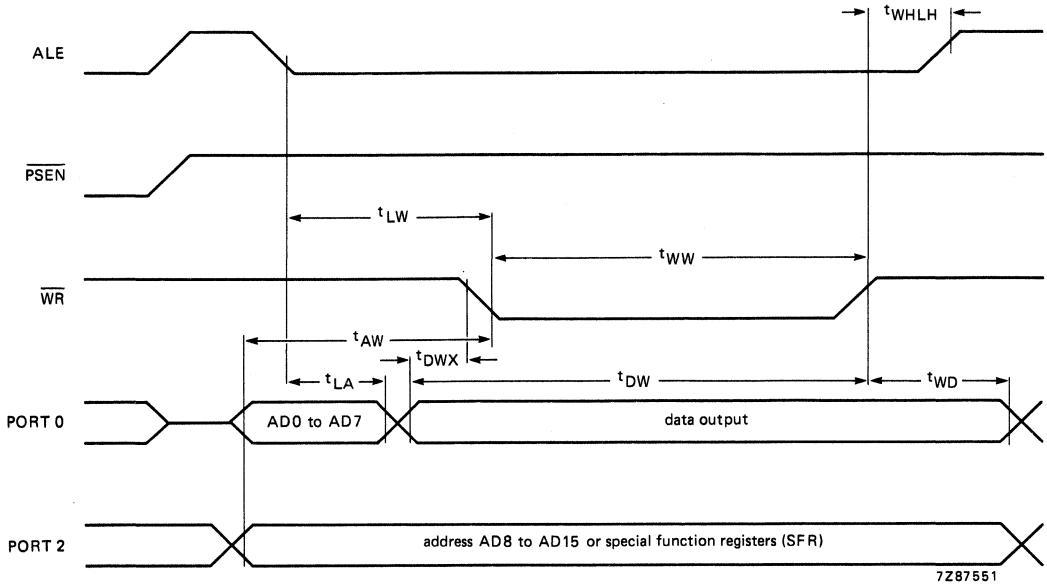


Fig. 16 Write to data memory.

DEVELOPMENT DATA

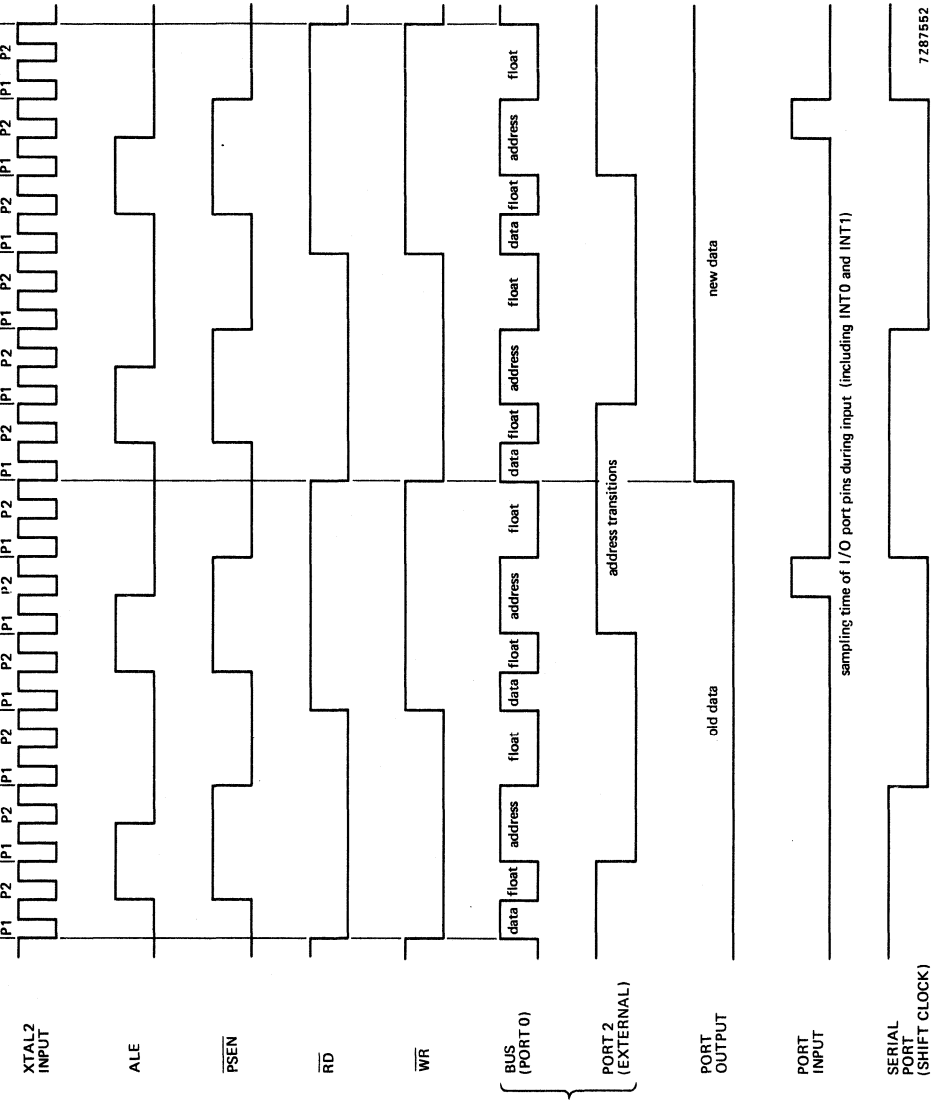


Fig. 17 Instruction cycle timing.

SINGLE-CHIP 8-BIT CMOS MICROCONTROLLER

DESCRIPTION

The PC80CXX family of single-chip 8-bit CMOS microcontrollers consists of:

- The PCB80C48 and PCB80C49 with resident mask programmed ROM.
- The PCB80C35 and PCB80C39 without resident program memory for use with external EPROM/ROM.

All versions are pin and function compatible to their NMOS counter parts but with additional features and high performance.

The PC80CXX family are designed to be efficient control processors as well as arithmetic processors. Their instruction set allows the user to directly set and reset individual I/O lines as well as test individual bits within the accumulator. A large variety of branch and table look-up instructions enable efficient implementation of standard logic functions. Code efficiency is high; over 70% of the instructions are single byte; all others are two byte.

An on-chip 8-bit counter is provided, which can count either machine cycles (÷ 32) or external events. The counter can be programmed to cause an interrupt to the processor.
Program and data memories can be expanded using standard devices. Input/output capabilities can be expanded using standard devices.
The family has low power consumption and in addition a power down mode is provided.

FEATURES

- 8-bit CPU, ROM, RAM, I/O in a single 40-pin package
- PCB80C48: 1K x 8 ROM, 64 x 8 RAM
- PCB80C49: 2K x 8 ROM, 128 x 8 RAM
- Internal counter/timer
- Internal oscillator, clock driver
- Single-level interrupts: external and counter/timer
- 17 internal registers: accumulator, 16 addressable registers
- Over 90 instructions: 70% single byte
- All instructions: 1 or 2 cycles
- Easily expandable memory and I/O
- TTL compatible inputs and outputs
- Single 5 V supply
- Wide frequency operating range
- Low current consumption
- Also available with extended temperature range; PCF 80CXX = −40 °C to + 85 °C

APPLICATIONS

- Peripheral interfaces and controllers
- Test and measurement instruments
- Sequencers
- Audio/video systems
- Environmental control systems
- Modems and data enciphering

PACKAGE OUTLINES

PCB/F80C35/C39/C48/C49P: 40-lead DIL; plastic (SOT-129).
PCB80C35/C39/C48/C49WP: 44-lead plastic leaded chip-carrier (PLCC); SOT-187A.

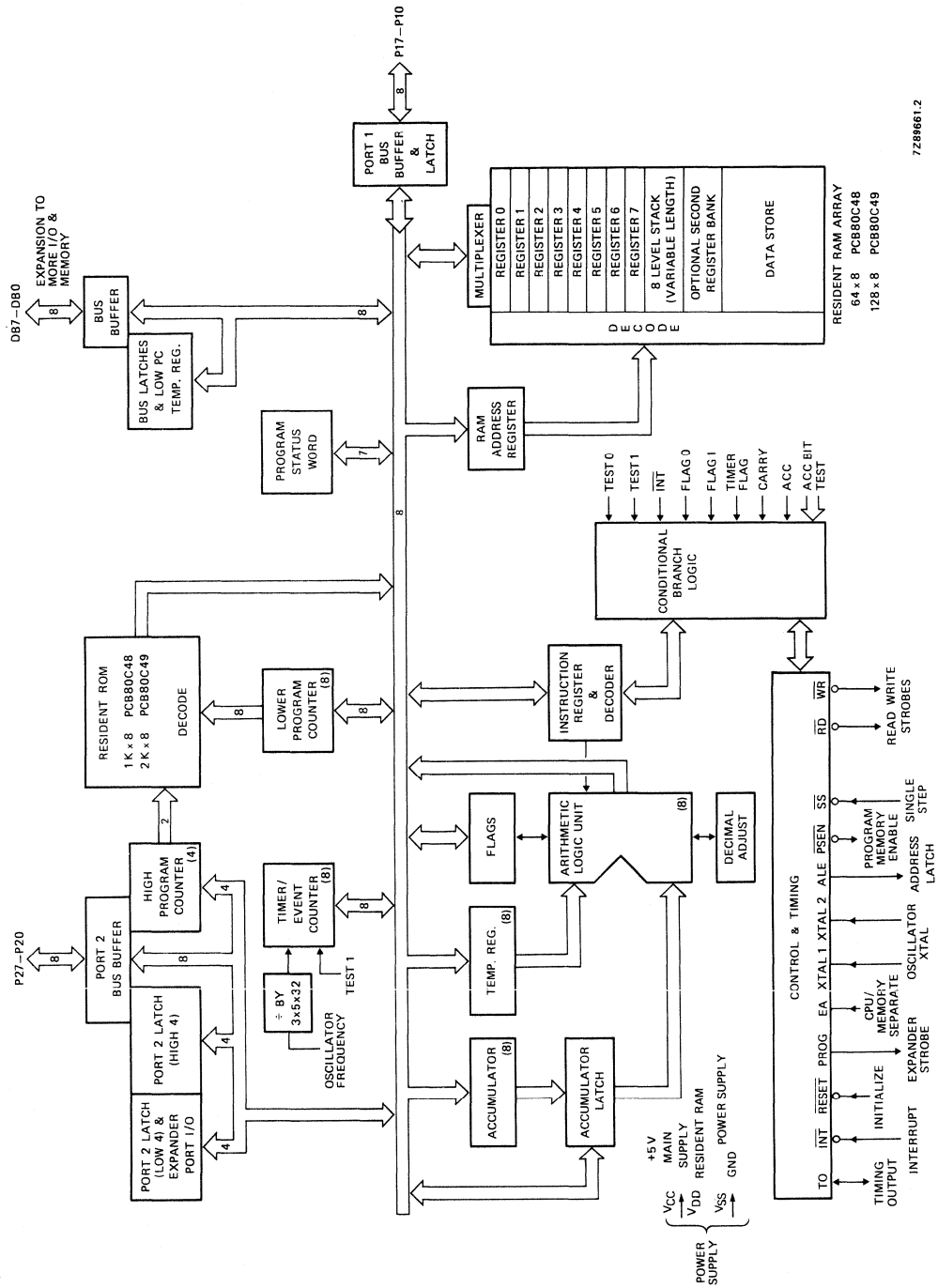


Fig. 1 Block diagram.

DEVELOPMENT DATA

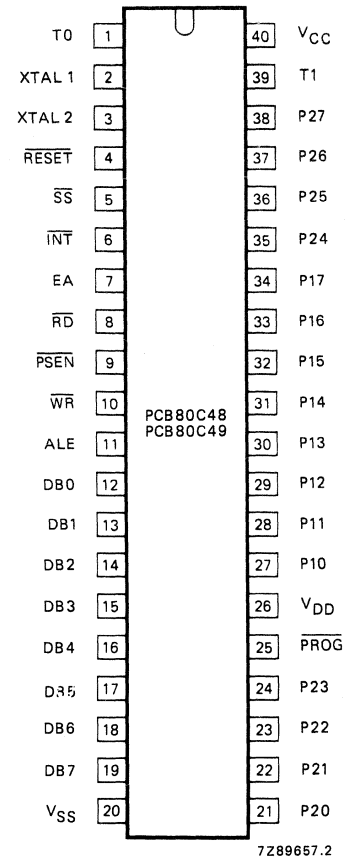


Fig. 2a Pinning diagram; for pin designation see next page.

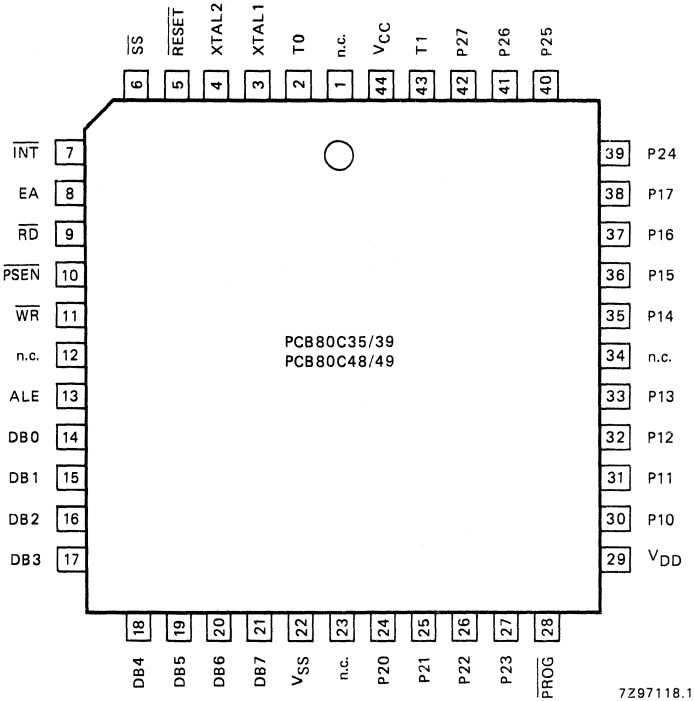


Fig. 2b Pinning diagram for PCB80CXXWP; for pin designation see next page.

Where: n.c. = not connected.

PIN DESIGNATION

designation	pin no.	function
DB0–DB7	12–19	**BUS.** Bidirectional I/O port that can be read or written using the $\overline{RD}$ and $\overline{WR}$ strobes. This port can also be statically latched. Contains the 8 lower order address bits during external memory access and receives the addressed instruction under control of $\overline{PSEN}$. $\overline{PSEN}$, ALE, $\overline{RD}$ and $\overline{WR}$ determine whether the access is an instruction fetch or a read/write access to external RAM.
P10–P17	27–34	**Port 1.** 8-bit quasi-bidirectional I/O port (note 1).
P20–P27	21–24, 35–38	**Port 2.** 8-bit quasi-bidirectional I/O port (note 1). P20–P23 contain the 4 higher order address bits during an access of external program memory.
$\overline{PROG}$	25	**Output strobe** (active LOW) for I/O expander.
T0	1	**Input pin** sensed using the JT0 and JNT0 instructions. **Clock output pin** when designated as such by the ENT0 CLK instruction.
T1	39	**Input pin** sensed using the JT1 and JNT1 instructions. Can be designated as the timer/counter input by the STRT CNT instruction.
$\overline{INT}$	6	**Interrupt input pin.** When LOW causes an interrupt in the current program if external interrupt is enabled. Can also be used as an input, testable using the JNI instruction. Interrupt is disabled during and after a RESET.
$\overline{RESET}$	4	**Reset input pin** used to initialize the microcontroller. Active LOW. During program verification the address is latched by a '0' to '1' transition on $\overline{RESET}$ and the data at the addressed location is output on BUS (note 2).
ALE	11	**Address latch enable.** Occurs once each machine cycle and is useful for timing and sampling. During external program or data memory access, ALE is used to strobe the address information multiplexed on the DB0 to DB7 outputs.
$\overline{RD}$	8	**Read BUS.** Active LOW strobe used to gate data onto BUS lines when reading from an external source.
$\overline{WR}$	10	**Write BUS.** Active LOW strobe used to write data from BUS lines to an external designation.
EA	7	**External access input.** When HIGH, forces instruction fetch from external memory.
$\overline{PSEN}$	9	**Program store enable.** Active LOW strobe that occurs only during a fetch from external program memory.
$\overline{SS}$	5	**Single step input.** Active LOW which is used with ALE to cause the microcontroller to execute a single instruction.
V_{DD}	26	**RAM power supply,** + 5 V during normal operation and power-down mode.

designation	pin no.	function
XTAL1	2	**One side of crystal** (or inductor) input for internal oscillator. Can also be used as an input for an external timing source (note 2).
XTAL2	3	**Other side of crystal.**
V_{SS}	20	**Ground.**
V_{CC}	40	**Main power supply**, + 5 V during normal operation.

Notes

1. Each port line can be designated as an input or an output. A line is designated as an input by first writing a logic '1' to the line. $\overline{\text{RESET}}$ sets all lines to logic '1'.
2. Non-standard TTL V_{IH}.

FUNCTIONAL DESCRIPTION

Detailed information available on request. See 8-bit microcontroller user manual.

Program memory

The resident program memory of each type is:
PCB80C48: 1024 byte ROM
PCB80C49: 2048 byte ROM
The PCB80C35 and PCB80C39 have no resident program memory.
The total addressing capability is 4096 bytes.
The program memory adress space is divided into two 2048-bytes banks MB0 and MB1 (Fig. 3).
Further, the program memory is divided into pages of 256 bytes for conditional branches.

There are three locations in program memory of special importance. These locations contain the first instruction to be executed after one of three events.
The three locations and their contents are:
location 0 – activation, then deactivation of the $\overline{\text{RESET}}$ line,
location 3 – activation of the $\overline{\text{INT}}$ line when the external interrupt is enabled,
location 7 – an overflow of the timer/counter if the T/C interrupt is enabled.

Data memory

The resident data memory, as shown in Fig. 4, is:
PCB80C48; PCB80C35: 64 byte RAM
PCB80C49; PCB80C39: 128 byte RAM
All locations are indirectly addressable by either of two RAM pointer registers at locations 0 and 1.
The first eight locations of RAM (0 to 7) are designated as working registers and are directly addressable by several instructions. By selecting register bank 1, RAM locations 24 to 31 become the working registers, replacing those in register bank 0 (0 to 7).
RAM locations 8 to 23 are designated as the stack. Two locations (bytes) are used per CALL, allowing up to eight levels of subroutine nesting.
If additional RAM is required, up to 256 bytes may be added and addressed directly using the MOVX instructions. If more RAM is required, an I/O port can be used to select one (256 byte) bank of external memory at a time.

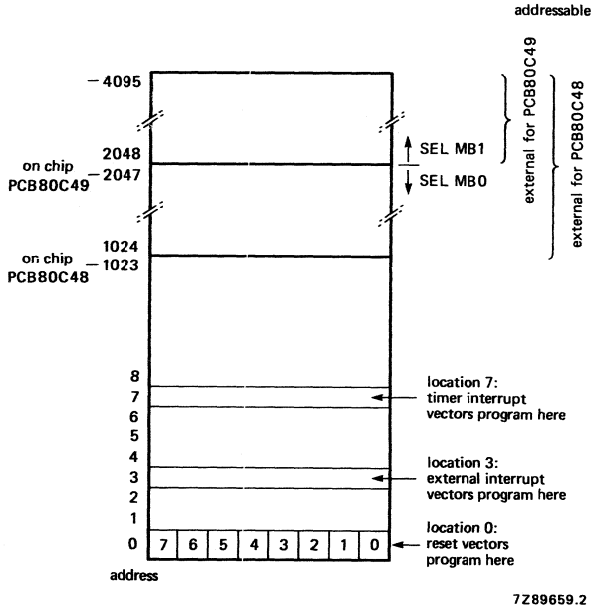


Fig. 3 Program memory map.

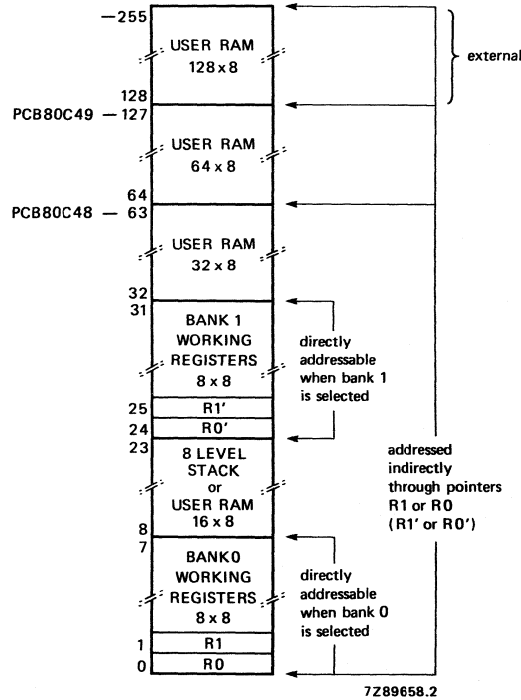


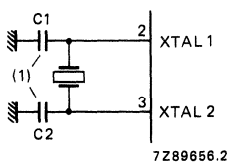
Fig. 4 Data memory map. In addition R0 or R1 (R0' or R1') may be used to address 256 words of an external RAM.

Program counter and stack

The program counter (PC) is a 12-bit counter/register that points to the location from which the next instruction is to be fetched. When EA is '0', the PC addresses an internal program memory. At the boundary of the internal program memory an automatic switch over to external memory is made. When EA is '1', all the program is fetched from external ROM/EPROM. The total address space is 4K bytes. An interrupt or CALL to a subroutine causes the contents of the program counter to be stored in one of the 8 register pairs of the program counter stack. The pair to be used is determined by a 3-bit stack pointer which is part of the program status word (PSW). Data RAM locations 8 to 23 are available as stack registers and are used to store the program counter and 4 bits of PSW. The stack pointer, when initialized to 000B, points to RAM locations 8 and 9. The first subroutine jump or interrupt results in the program counter contents being transferred to locations 8 and 9 of the RAM array. The stack pointer is then incremented by one to point to locations 10 and 11 in anticipation of another CALL. Nesting of subroutines within subroutines can continue up to 8 times without overflowing the stack. If overflow does occur the deepest address stored (location 8 and 9) will be overwritten and lost since the stack pointer overflows from 111 to 000. It also underflows from 000 to 111.
The end of a subroutine, which is signalled by a return instruction (RET or RETR), causes the stack pointer to be decremented and the contents of the resulting register pair to be transferred to the program counter.

Oscillator and clock

The PCB80C48 and PCB80C49 both contain their own internal oscillator and clock driver.
A crystal, inductor or external pulse generator determines the oscillator frequency (see Figs 5, 6 and 7).
The output of the oscillator is divided-by-three and is available at T0 (pin 1) by executing the ENT0 CLK instruction. This CLK signal is divided-by-five to define a machine (instruction) cycle. It is available at ALE (pin 11).



For quartz crystal
1 to 11 MHz: C1 = C2 = 15 to 25 pF
For ceramic resonators
1 to 11 MHz: C1 = C2 = 30^{+5}_{-10} pF

(1) Including crystal-socket stray capacitance.

Fig. 5 Crystal oscillator mode. Typical values are given. Crystal serial impedance should be $< 75\ \Omega$ at 6 MHz and $< 180\ \Omega$ at 3,6 MHz.

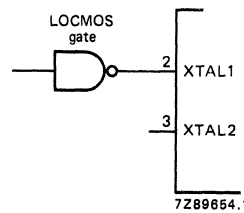
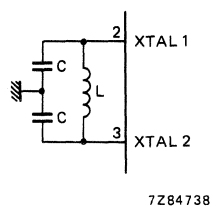


Fig. 6 Driving from an external source. Test conditions at XTAL1; Minimum HIGH (> 0,7 of V_{CC}) and LOW (< 0,13 of V_{CC}) times, should be at least 45% of a clock period.



L (μH)	C (pF)	f_{nom} (MHz)
45	20	5,2
120	20	3,2

Fig. 7 LC oscillator mode.

FUNCTIONAL DESCRIPTION (continued)

Timer/event counter

An internal counter is available which can count either external events or machine cycles (÷ 32). The machine cycles are divided-by-32 before they are applied to the input of the 8-bit counter. External events are applied directly to the input of the counter. The maximum frequency that can be counted is one third of the machine cycle frequency. The minimum positive duty cycle that can be detected is 0,2 times the cycle period. The counter is under program control and can be made to generate an interrupt to the processor when it overflows.

Interrupt

An interrupt may be generated by either an external input ($\overline{INT}$, pin 6) or the overflow of the internal timer/event counter, when enabled. In either case, the processor completes execution of the present instruction and then does a CALL to the interrupt service routine, After service, a RETR instruction restores the machine to the state it was prior to the interrupt. The external interrupt has priority over the internal interrupt.

Input/output

The PC80CXX family has 27 I/O lines. These lines are arranged as three 8-line ports, which serve as either inputs, outputs or bidirectional ports and 3 'test' inputs which can alter program sequences when tested by conditional jump instructions.

Ports 1 and 2

Ports 1 and 2 are each 8-bits wide and have identical characteristics. Data written to these ports are statically latched and remains unchanged until rewritten. As input ports these lines are non-latching, e.g., inputs must be present until read by an input instruction. Inputs are fully TTL compatible and outputs will drive one standard TTL load.

The lines of ports 1 and 2 are called quasi-bidirectional because of a special output circuit structure which allows each line to serve as an input, an output, or both even though outputs are statically latched. The circuit configuration is shown in Fig. 8. Each line has a unique high-impedance pull-up transistor TR3, this is turned on when the line is pulled above 2 V by an external source or by writing a logic '1' to the port. This pull-up is sufficient to provide the source current for a TTL HIGH level, yet can be pulled LOW by a standard TTL gate, thus allowing the same pin to used for both input and output. When a logic '1' is written to a line, a second high impedance transistor TR2, pulls the line up to 5 V. To provide fast switching during a '0' to '1' transition, a relatively low-impedance transistor TR1 (approx. 750 Ω) is switched on for 1/5 of a machine cycle whenever a '1' is written to the line. Whenever a '0' is written to the line, a low-impedance (approx. 250 Ω) transistor TR4, overcomes the light pull-up and provides TTL current sinking capability.

Since the pull-down transistor TR4 is a low-impedance device, a '1' must first be written to any line which is to be used as an input. RESET initializes all lines to the high impedance '1' state. This structure allows input and output on the same pin and also allows a mixture of input lines and output lines on the same port. The quasi-bidirectional port in combination with the ANL and ORL logical instructions provide an efficient means for handling single line inputs and outputs within an 8-bit processor.

BUS

BUS is also an 8-bit port which is a true bidirectional port with associated input and output strobes. If the bidirectional feature is not needed, BUS can serve as either a statically latched output port or non-latching input port. Input and output lines on this port cannot be mixed.
As a static port, data is written and latched using the OUTL instruction and input using the INS instruction. The INS and OUTL instructions generate pulses on the corresponding $\overline{RD}$ and $\overline{WR}$ output strobe lines; however, in the static port mode they are generally not used. As a bidirectional port, the MOVX instructions are used to read and write the port. A write to the port generates a pulse on the $\overline{WR}$ output line and output data is valid at the trailing-edge of $\overline{WR}$. A read of the port generates a pulse on the $\overline{RD}$ output line and input data must be valid at the trailing-edge of $\overline{RD}$. When not being written or read, the BUS lines are high impedance.

Test (T0, T1) and $\overline{INT}$ inputs

Three pins serve as inputs and are testable with the conditional jump instruction. These pins are T0, T1, and $\overline{INT}$ and they allow inputs to cause program branches without the necessity to load an input port into the accumulator. The T0, T1 and $\overline{INT}$ pins have other possible functions as well.

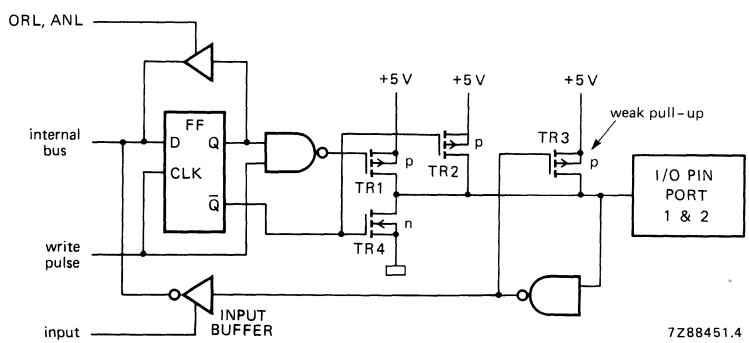


Fig. 8 Quasi-bidirectional port structure.

$\overline{RESET}$ input

The $\overline{RESET}$ input provides a means for initialization for the processor. This Schmitt-trigger input has an internal pull-up resistor. The combination of an external 47 kΩ resistor and a 1 μF capacitor provides a reset pulse of sufficient duration to guarantee that all circuitry is reset. If the reset pulse is generated otherwise, the $\overline{RESET}$ pin must be held at ground for at least 10 ms after the (0,13 V_{CC}) power supply is within tolerance. Only five machine cycles (2,5 μs at 6 MHz) are required if power is already on and the oscillator has stabilized. Typical circuitry is shown in Fig. 10.

Single step input ($\overline{SS}$)

By proper control of the $\overline{SS}$ line, the processor can be forced to execute one instruction and then to wait until the single step switch is activated again.

FUNCTIONAL DESCRIPTION (continued)

IDLE mode

The PC80CXX family is provided with a IDLE mode in which the internal oscillator, the internal timer and the external interrupt and counter are still functioning, while the status of following parts is maintained: RAM and register/Port 1 and 2/Bus. The IDLE mode is entered after execution of the IDLE instruction (opcode 1H). The IDLE mode is terminated by one of the two possible interrupts, if enabled, or a RESET signal. If an external interrupt terminates the IDLE mode, the next instruction that is executed is at location 3 of the program store. In case a timer/counter interrupt terminates the IDLE mode, the next instruction is at location 7. The reset signal will not only terminate the IDLE mode, but will also initialize the processor.

Power-down mode

In the PC80CXX family, power can be removed from all but the 64 x 8 bit(80C48) and 128 x 8 bit (80C49) data RAM array, for low power standby operation. In the power-down mode the contents of the data RAM can be maintained. V_{CC} serves as the + 5 V supply pin for the bulk of the circuitry, while the V_{DD} pin supplies only the RAM array. In normal operation, both pins are at + 5 V. In standby, V_{CC} is at ground and only V_{DD} is maintained at + 5 V.

Applying $\overline{\text{RESET}}$ to the processor through the $\overline{\text{RESET}}$ pin inhibits any access to the RAM by the processor and guarantees that the RAM cannot be inadvertently altered as power is removed from V_{CC}. A typical power-down sequence occurs as shown in Fig. 9.

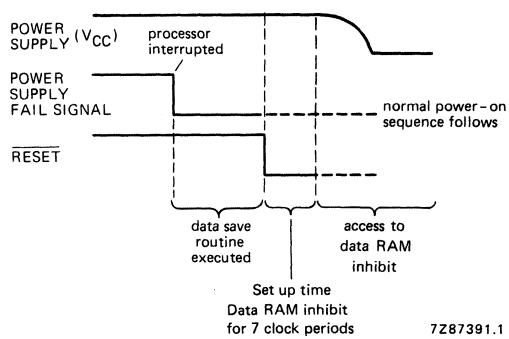


Fig. 9 Power-down sequence.

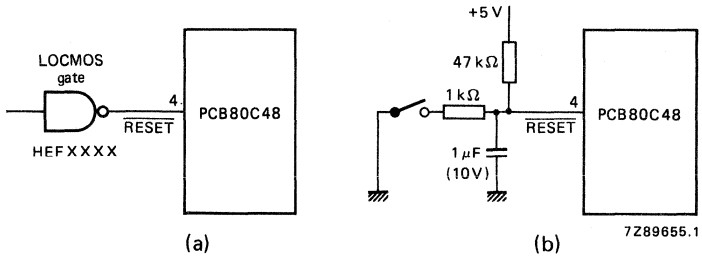


Fig. 10 An external reset circuit is shown in (a) and power-on reset in (b).

DEVELOPMENT DATA

Instruction set

The PC80CXX instruction set consists of over 90 one and two-byte instructions (see Table 1). Program code efficiency is high because:

- working registers and program variables are stored in the RAM, which require only a single byte to address,
- program memory is divided into pages of 256 bytes, which means that branch destination addresses require one byte.

The instruction set efficiently manipulates and tests bits in addition to performing logical and arithmetic operations upon and the testing of bytes. A set of MOVE instructions operates indirectly upon either RAM or ROM, which permits efficient access of pointers and data tables. The indirect jump instruction performs a multi-way branch (up to 256) upon the content of the accumulator to addresses stored in a look-up table. The 'decrement register and jump if not zero' instruction saves a byte every time it is used as opposed to using separate increment and test instructions.

The on-chip counter enables either external events or time to be counted off-line from the main program. The PC80CXX can either test the counter (under program control) or cause its overflow to generate an interrupt. These features are highly desirable for real-time applications. See Table 2 for instruction timing.

Table 1 Instruction set is shown on the next 7 pages.

Symbol definitions used in Table 1.

symbol	description
A	the accumulator
AC	the auxiliary carry flag
addr	program memory address (11-bits)
Bb	bit designation (b = 0–7)
BS	the bank switch
C	carry flag
CLK	clock signal
CNT	event counter
D	nibble designation (4-bits)
DBF	program memory bank flip-flop
data	number or expression (8-bits)
F0, F1	flags 0 and 1
$\overline{\text{I}}$	interrupt
$\overline{\text{INT}}$	external interrupt
P	'in-page' operation designation
Pp	port designation (p = 1, 2 or 4–7)
PSW	program status word
Rr	register designation (r = 0, 1 or 0–7)
SP	stack pointer
T	timer
TF	timer flag
T0, T1,	test 0 and 1 inputs
#	immediate data prefix
@	indirect address prefix
$	current value of program counter
←	is replaced by
↔	is exchanged with

Notes to Table 1.

1. Instruction code designations r and p form the binary representation of the registers and ports involved.
2. The dot under the appropriate flag bit indicates that its content is subject to change by the instruction it appears in.
3. Numerical subscripts appearing in the FUNCTION column reference the specific bits affected.

	mnemonic	function	description	instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	cycles	bytes	flags C	AC	F0	F1	BS
ACCUMULATOR	ADD A,Rr	(A)←(A) + (Rr) for r = 0–7	Add contents of designated register to A	0 1 1 0 1 r r r	1	1	•	•			
	ADD A,@Rr	(A)←(A) + ((Rr)) for r = 0–1	Add indirect the contents of the data memory location to A	0 1 1 0 0 0 0 r	1	1	•	•			
	ADD A,#data	(A)←(A) + data	Add immediate data to A	0 0 0 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
	ADDC A,Rr	(A)←(A) + (C) + (Rr) for r = 0–7	Add with carry the contents of designated register to A	0 1 1 1 1 r r r	1	1	•	•			
	ADDC A,@Rr	(A)←(A) + (C) + ((Rr)) for r = 0–1	Add indirect with carry the contents of the data memory location to A	0 1 1 1 0 0 0 r	1	1	•	•			
	ADDC A,#data	(A)←(A) + (C) + data	Add immediate data with carry to A	0 0 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2	•	•			
	ANL A,Rr	(A)←(A) AND (Rr) for r = 0–7	Logical AND contents of designated register with A	0 1 0 1 1 r r r	1	1					
	ANL A,@Rr	(A)←(A) AND ((Rr)) for r = 0–1	Logical AND indirect the contents of data memory with A	0 1 0 1 0 0 0 r	1	1					
	ANL A,#data	(A)←(A) AND data	Logical AND immediate data with A	0 1 0 1 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
	CLR A	(A)←0	Clear the contents of A	0 0 1 0 0 1 1 1	1	1					
	CPL A	(A)←NOT(A)	Complement the contents of A	0 0 1 1 0 1 1 1	1	1					
	DA A		Decimal adjust the contents of A	0 1 0 1 0 1 1 1	1	1	•	•			
	DEC A	(A)←(A) –1	Decrement the contents of A by 1	0 0 0 0 0 1 1 1	1	1					
	INC A	(A)←(A) + 1	Increment the contents of A by 1	0 0 0 1 0 1 1 1	1	1					
	ORL A,Rr	(A)←(A) OR (Rr) for r = 0–7	Logical OR contents of designated register with A	0 1 0 0 1 r r r	1	1					
	ORL A,@Rr	(A)←(A) OR ((Rr)) for r = 0–1	Logical OR indirect the contents of data memory location with A	0 1 0 0 0 0 0 r	1	1					

ACCUMULATOR (continued)	ORL A,#data	(A)←(A) OR data	Logical OR immediate data with A	0 d_7	1 d_6	0 d_5	0 d_4	0 d_3	0 d_2	1 d_1	1 d_0	2	2			
	RL A	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(A_7)$ for n = 0–6	Rotate A left by 1-bit without carry	1	1	1	0	0	1	1	1	1	1			
	RLC A	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(C)$ $(C)\leftarrow(A_7)$ for n = 0–6	Rotate A left by 1-bit through carry	1	1	1	1	0	1	1	1	1	1	•		
	RR A	$(A_n)\leftarrow(A_{n+1})$ $(A_7)\leftarrow(A_0)$ for n = 0–6	Rotate A right by 1-bit without carry	0	1	1	1	0	1	1	1	1	1			
	RRC A	$(A_n)\leftarrow(A_{n+1})$ $(A_7)\leftarrow(C)$ $(C)\leftarrow(A_0)$ n = 0–6	Rotate A right by 1-bit through carry	0	1	1	0	0	1	1	1	1	1	•		
	SWAP A	$(A_{4-7})\leftrightarrow(A_{0-3})$	Swap the two 4-bit nibbles in A	0	1	0	0	0	1	1	1	1	1			
	XRL A,Rr	(A)←(A) XOR (Rr)	Logical XOR contents of designated register with A	1	1	0	1	1	r	r	r	1	1			
	XRL A,@Rr	(A)←(A) XOR ((Rr))	Logical XOR indirect the contents of data memory location with A	1	1	0	1	0	0	0	r	1	1			
	XRL A,#data	(A)←(A) XOR data	Logical XOR immediate data with A	1 d_7	1 d_6	0 d_5	1 d_4	0 d_3	0 d_2	1 d_1	1 d_0	2	2			

	mnemonic	function	description	instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	cycles	bytes	flags C	AC	F0	F1	BS
BRANCH	DJNZ Rr, addr	(Rr)←(Rr) −1 if (Rr) not zero: (PC_{0-7})←addr	Decrement the specified register and test contents	1 1 1 0 1 r r r a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JBb addr	(PC_{0-7})←addr if Bb = 1; (PC)←(PC) + 2 if Bb = 0	Jump to specified address if accumulator bit is set	b_2 b_1 b_0 1 0 0 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JC addr	(PC_{0-7})←addr if C = 1; (PC)←(PC) + 2 if C = 0	Jump to specified address if carry flag is set	1 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JF0 addr	(PC_{0-7})←addr if (F0 = 1; (PC)←(PC) + 2 if F0 = 0	Jump to specified address if flag F0 is set	1 0 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JF1 addr	(PC_{0-7})←addr if F1 = 1; (PC)←(PC) + 2 if F1 = 0	Jump to specified address if flag F1 is set	0 1 1 1 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JMP addr	(PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11})←(DBF)	Direct jump to specified address within the 2K address block	a_{10} a_9 a_8 0 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JMPP @A	(PC_{0-7})←((A))	Jump indirect to specified address within address page	1 0 1 1 0 0 1 1	2	1					
	JNC addr	(PC_{0-7})←addr if C = 0; (PC)←(PC) + 2 if C = 1	Jump to specified address if carry flag is LOW	1 1 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JNI	(PC_{0-7})←addr if $\overline{INT}$ = 0; (PC)←(PC) + 2 if $\overline{INT}$ = 1	Jump to specified address if $\overline{INT}$ input is LOW	1 0 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JNT0 addr	(PC_{0-7})←addr if T0 = 0; (PC)←(PC) + 2 if T0 = 1	Jump to specified address if T0 is LOW	0 0 1 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	JNT1 addr	(PC_{0-7})←addr if T1 = 0; (PC)←(PC) + 2 if T1 = 1	Jump to specified address if T1 is LOW	0 1 0 0 0 1 1 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					

BRANCH (continued)	JNZ addr	$(PC_{0-7}) \leftarrow$addr if $A \neq 0$; $(PC) \leftarrow (PC) + 2$ if $A = 0$	Jump to specified address if A is non-zero	1 a_7	0 a_6	0 a_5	1 a_4	0 a_3	1 a_2	1 a_1	0 a_0	2	2					
	JTF addr	$(PC_{0-7}) \leftarrow$addr if TF = 1; $(PC) \leftarrow (PC) + 2$ if TF = 0	Jump to specified address if timer flag is set to 1	0 a_7	0 a_6	0 a_5	1 a_4	0 a_3	1 a_2	1 a_1	0 a_0	2	2					
	JT0 addr	$(PC_{0-7}) \leftarrow$addr if T0 = 1; $(PC) \leftarrow (PC) + 2$ if T0 = 0	Jump to specified address if T0 = 1	0 a_7	0 a_6	1 a_5	1 a_4	0 a_3	1 a_2	1 a_1	0 a_0	2	2					
	JT1 addr	$(PC_{0-7}) \leftarrow$addr if T1 = 1; $(PC) \leftarrow (PC) + 2$ if T1 = 0	Jump to specified address if T1 = 1	0 a_7	1 a_6	0 a_5	1 a_4	0 a_3	1 a_2	1 a_1	0 a_0	2	2					
	JZ addr	$(PC_{0-7}) \leftarrow$addr if $A = 0$; $(PC) \leftarrow (PC) + 2$ if $A \neq 0$	Jump to specified address if A is zero	1 a_7	1 a_6	0 a_5	0 a_4	0 a_3	1 a_2	1 a_1	0 a_0	2	2					
CONTROL	EN I		Enable external ($\overline{INT}$) interrupt	0	0	0	0	0	1	0	1	1	1					
	DIS I		Disable external ($\overline{INT}$) interrupt	0	0	0	1	0	1	0	1	1	1					
	SEL RB0	(BS)←0	Select bank 0 (locations 0–7) of data memory	1	1	0	0	0	1	0	1	1	1					•
	SEL RB1	(BS)←1	Select bank 1 (locations 24–31) of data memory	1	1	0	1	0	1	0	1	1	1					•
	SEL MB0	(DBF)←0	Select program memory bank 0; addresses 0–2047	1	1	1	0	0	1	0	1	1	1					
	SEL MB1	(DBF)←1	Select program memory bank 1; addresses 2048–4095	1	1	1	1	0	1	0	1	1	1					
	ENT0 CLK		Enable clock output onto T0	0	1	1	1	0	1	0	1	1	1					

	mnemonic	function	description	instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	cycles	bytes	flags C	AC	F0	F1	BS
DATA MOVES	MOV A,#data	(A)←data	Move immediate data into A	0 0 1 0 0 0 1 1 d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
	MOV A,Rr	(A)←(Rr) for r = 0–7	Move the contents of the designated register into A	1 1 1 1 1 r r r	1	1					
	MOV A,@Rr	(A)←((R)) for r = 0–1	Move indirect the contents of data memory into A	1 1 1 1 0 0 0 r	1	1					
	MOV A,PSW	(A)←(PSW)	Move contents of the program status word into A	1 1 0 0 0 1 1 1	1	1					
	MOV Rr,#data	(Rr)←data for r = 0–7	Move immediate data into the designated register	1 0 1 1 1 r r r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
	MOV Rr,A	(Rr)←(A) for r = 0–7	Move A contents into the designated register	1 0 1 0 1 r r r	1	1					
	MOV @Rr,A	((Rr))←(A) for r = 0–1	Move indirect A contents into data memory location	1 0 1 0 0 0 0 r	1	1					
	MOV @Rr,#data	((Rr))←data for r = 0–1	Move indirect the specified data into data memory	1 0 1 1 0 0 0 r d_7 d_6 d_5 d_4 d_3 d_2 d_1 d_0	2	2					
	MOV PSW,A	(PSW)←A	Move contents of A into the program status word	1 1 0 1 0 1 1 1	1	1	•	•	•		•
	MOVP A,@A	(A)←((A))	Move data in the current page into A	1 0 1 0 0 0 1 1	2	1					
	MOVP3 A,@A	(A)←((A)) in page 3	Move data in page 3 of memory bank Ø into A	1 1 1 0 0 0 1 1	2	1					
	MOVX A,@Rr	(A)←((R)) for r = 0–1	Move indirect the contents of external memory location into A	1 0 0 0 0 0 0 r	2	1					
	MOVX @Rr,A	((Rr))←(A) for r = 0–1	Move indirect the contents of A into external memory	1 0 0 1 0 0 0 r	2	1					
	XCH A,Rr	(A)↔(Rr) for r = 0–7	Exchange A with designated register contents	0 0 1 0 1 r r r	1	1					
	XCH A,@Rr	(A)↔((R)) for r = 0–1	Exchange indirect A contents with location in data memory	0 0 1 0 0 0 0 r	1	1					

DEVELOPMENT DATA

D.M.	XCHD A,@Rr	$(A_{0-3}) \leftrightarrow (Rr_{0-3})$ for r = 0–1	Exchange indirect 4-bit contents of A with data memory	0	0	1	1	0	0	0	r	1	1					
FLAGS	CPL C	(C)←NOT (C)	Complement content of carry bit	1	0	1	0	0	1	1	1	1	1	•				
	CPL F0	(F0)←NOT (F0)	Complement content of flag F0	1	0	0	1	0	1	0	1	1	1			•		
	CPL F1	(F1)←NOT (F1)	Complement content of flag F1	1	0	1	1	0	1	0	1	1	1				•	
	CLR C	(C)←0	Clear content of carry bit to 0	1	0	0	1	0	1	1	1	1	1	•				
	CLR F0	(F0)←0	Clear content of flag F0 to 0	1	0	0	0	0	1	0	1	1	1			•		
	CLR F1	(F1)←0	Clear content of flag F1 to 0	1	0	1	0	0	1	0	1	1	1				•	
INPUT/OUTPUT	ANL BUS,#data	(BUS)←(BUS) AND data	Logical AND immediate data with BUS	1	0	0	1	1	0	0	0	2	2					
				d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0							
	ANL Pp,#data	(Pp)←(Pp) AND data; p = 1–2	Logical AND immediate data with designated port (1 or 2)	1	0	0	1	1	0	p	p	2	2					
				d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0							
	ANLD Pp,A	(Pp)←(Pp) AND (A_{0-3}); p = 4–7	Logical AND contents of A with designated port (4–7)	1	0	0	1	1	1	p	p	2	1					
	IN A,Pp	(A)←(Pp) p = 1–2	Input data from designated port (1–2) into A	0	0	0	0	1	0	p	p	2	1					
	INS A,BUS	(A)←(BUS)	Input strobed BUS data into A	0	0	0	0	1	0	0	0	1	2					
	MOVD A,Pp	(A_{0-3})←(Pp); p = 4–7 (A_{4-7})←0	Move contents of designated port (4–7) into A	0	0	0	0	1	1	p	p	2	1					
	MOVD Pp,A	(Pp)←(A_{0-3}) p = 4–7	Move contents of A to designated port (4–7)	0	0	1	1	1	1	p	p	1	1					
	ORLD Pp,A	(Pp)←(Pp) OR (A_{0-3}); p = 4–7	Logical OR contents of A with designated port (4–7)	1	0	0	0	1	1	p	p	1	1					
	ORL BUS,#data	(BUS)←(BUS) OR data	Logical OR immediate data with BUS	1	0	0	0	1	0	0	0	2	2					
				d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0							
	ORL Pp,#data	(Pp)←(Pp) OR data p = 1–2	Logical OR immediate data with designated port (1–2)	1	0	0	0	1	0	p	p	2	2					
				d_7	d_6	d_5	d_4	d_3	d_2	d_1	d_0							
	OUTL BUS,A	(BUS)←(A)	Output contents A onto BUS	0	0	0	0	0	0	1	0	1	2					
	OUTL Pp,A	(Pp)←(A) p = 1–2	Output contents A to designated port (1–2)	0	0	1	1	1	0	p	p	1	1					

	mnemonic	function	description	instruction code D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0	cycles	bytes	flags C	AC	F0	F1	BS
REGISTER	DEC Rr	(Rr)←(Rr) −1 for r = 0–7	Decrement contents of designated register by 1	1 1 0 0 1 r r r	1	1					
	INC Rr	(Rr)←(Rr) + 1 for r = 0–7	Increment contents of designated register by 1	0 0 0 1 1 r r r	1	1					
	INC @Rr	((R))←((Rr)) + 1 for r = 0–1	Increment indirect the contents of data memory location by 1	0 0 0 1 0 0 0 r	1	1					
SUBROUTINE	CALL addr	((SP))←(PC), (PSW_{4-7}) (SP)←(SP) + 1 (PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$	Call designated subroutine	a_{10} a_9 a_8 1 0 1 0 0 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0	2	2					
	RET	(SP)←(SP) −1 (PC)←((SP))	Return from subroutine without restoring program status word	1 0 0 0 0 0 1 1	2	1					
	RETR	(SP)←(SP) −1 (PC)←((SP)) (PSW_{4-7})←((SP))	Return from subroutine restoring program status word	1 0 0 1 0 0 1 1	2	1					
TIMER/COUNTER	EN TCNTI		Enable timer/counter interrupt	0 0 1 0 0 1 0 1	1	1					
	DIS TCNTI		Disable timer/counter interrupt	0 0 1 1 0 1 0 1	1	1					
	MOV A,T	(A)←(T)	Move contents of timer/counter into A	0 1 0 0 0 0 1 0	1	1					
	MOV T,A	(T)←(A)	Move contents of A into timer/counter	0 1 1 0 0 0 1 0	1	1					
	STOP TCNT		Stop count for event counter or timer	0 1 1 0 0 1 0 1	1	1					
	STRT CNT		Start count for event counter	0 1 0 0 0 1 0 1	1	1					
	STRT T		Start count for timer	0 1 0 1 0 1 0 1	1	1					
	IDLE		Entering IDLE mode	0 0 0 0 0 0 0 1	1	1					
	NOP		No operation	0 0 0 0 0 0 0 0	1	1					

Table 2 Instruction timing (see also Figs 11 and 12)

	cycle 1					cycle 2				
instruction	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
IN A,P	fetch instruction	increment program counter	–	increment timer	–	–	read port	* –	–	–
OUTL P,A	|	|	–	|	output to port	–	–	* –	–	–
ANL P,#data	|	* |	–	|	read port	fetch immediate data	–	*increment program counter	output to port	–
ORL P,#data	|	* |	–	|	read port	fetch immediate data	–	*increment program counter	output to port	–
INS A,BUS	|	|	–	|	–	–	read port	* –	–	–
OUTL BUS,A	|	|	–	|	output to port	–	–	* –	–	–
ANL BUS,#data	|	* |	–	|	read port	fetch immediate data	–	*increment program counter	output to port	–
ORL BUS,#data	|	* |	–	|	read port	fetch immediate data	–	*increment program counter	output to port	–
MOVX @R,A	|	|	output RAM address	|	output data to RAM	–	–	* –	–	–
MOVX A,@R	|	|	output RAM address	|	–	–	read data	* –	–	–
MOVD A,P	| fetch instruction	increment program counter	output opcode/ address	increment timer	–	–	read P2 lower	* –	–	–

Continued on next page.

Table 2 Instruction timing (continued)

	cycle 1					cycle 2				
instruction	S1	S2	S3	S4	S5	S1	S2	S3	S4	S5
MOVD P,A	fetch instruction	increment program counter	output opcode/ address	increment timer	output data to P2 lower	–	–	* –	–	–
ANLD P,A	|		output opcode/ address	|	output data	–	–	* –	–	–
ORLD P,A	|		output opcode/ address	|	output data	–	–	* –	–	–
J (conditional)	|	*	sample condition	increment timer	–	fetch immediate data	–	*update program counter	–	–
STRT CNT STRT T	|	*	–	–	start counter					
STOP TCNT	|	*	–	–	stop counter					
EN I	|	*	–	enable interrupt	–					
DIS I	|	*	–	disable interrupt	–					
ENT0 CLK	fetch instruction	*increment program counter	–	enable clock	–					

* Valid instruction addresses are output at this time if external program memory is being accessed.

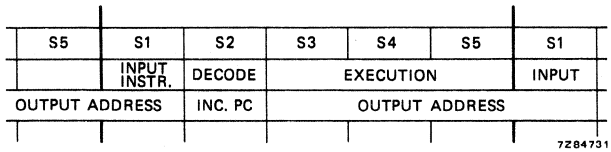


Fig. 11 Instruction cycle.

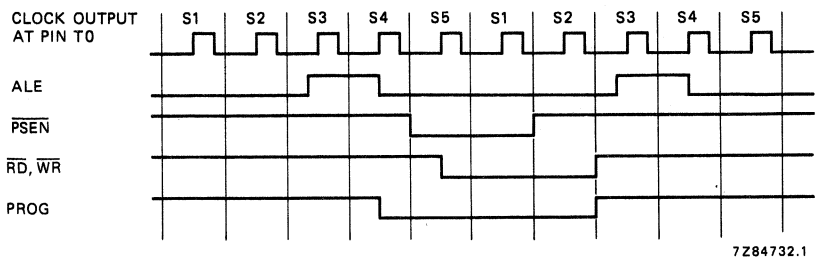


Fig. 12 Instruction cycle timing.

DEVELOPMENT DATA

Table 4 Instruction map.

first hexadicimal character of opcode	second hexadicimal character of opcode														
	0 1	2	3	4	5	6	7	8	9 1 2	A	B	C 4 5 6 7	D	E	F
0	NOP IDLE mode	OUTL BUS,A	ADD A, #data	JMP page 0	EN I		DEC A	INS A,BUS	IN A, Pp			MOVD A,Pp			
1	INC @Rr 0 1	JB0 addr	ADDC A, #data	CALL page 0	DIS I	JTF addr	INC A	INC Rr 0 1 2 3 4 5 6 7							
2	XCH A, @Rr 0 1		MOV A, #data	JMP page 1	EN TCNTI	JNT0 addr	CLR A	XCH A,Rr 0 1 2 3 4 5 6 7							
3	XCHD A, @Rr 0 1	JB1 addr		CALL page 1	DIS TCNTI	JT0 addr	CPL A		OUTL Pp,A			MOVD Pp,A			
4	ORL A, @Rr 0 1	MOV A, T	ORL A, #data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	ORL A,Rr 0 1 2 3 4 5 6 7							
5	ANL A, @Rr 0 1	JB2 addr	ANL A, #data	CALL page 2	STRT T	JT1 addr	DA, A	ANL A,Rr 0 1 2 3 4 5 6 7							
6	ADD A, @Rr 0 1	MOV T,A		JMP page 3	STOP TCNT		RRC A	ADD A,Rr 0 1 2 3 4 5 6 7							
7	ADDC A, @Rr 0 1	JB3 addr		CALL page 3	ENT0 CLK	JF1 addr	RR A	ADDC A,Rr 0 1 2 3 4 5 6 7							
8	MOVX A, @Rr 0 1		RET	JMP page 4	CLR F0	JN1 addr		ORL BUS, #data	ORL Pp, #data			ORLD Pp,A			
9	MOVX @Rr,A 0 1	JB4 addr	RETR	CALL page 4	CPL F0	JNZ addr	CLR C	ANL BUS, #data	ANP Pp, #data			ANLD Pp,A			
A	MOV @Rr, A 0 1		MOVP A, @A	JMP page 5	CLR F1		CPL C	MOV Rr,A 0 1 2 3 4 5 6 7							
B	MOV @Rr, #data 0 1	JB5 addr	JMPP @A	CALL page 5	CPL F1	JF0 addr		MOV R, #data 0 1 2 3 4 5 6 7							
C				JMP page 6	SEL RB0	JZ addr	MOV A, PSW	DEC Rr 0 1 2 3 4 5 6 7							
D	XRL A, @Rr 0 1	JB6 addr	XRL A, #data	CALL page 6	SEL RB1		MOV PSW, A	XRL A,Rr 0 1 2 3 4 5 6 7							
E			MOVP3 A @A	JMP page 7	SEL MB0	JNC addr	RL A	DJNZ Rr, addr 0 1 2 3 4 5 6 7							
F	MOV A, @Rr 0 1	JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	MOV A,Rr 0 1 2 3 4 5 6 7							

DEVELOPMENT DATA

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

D.C. current into any input or output	$\pm I_I, \pm I_O$	max.	10 mA
Total power dissipation	P_{tot}	max.	1,5 W
Storage temperature range	T_{stg}		−65 to + 150 °C
Operating ambient temperature range	T_{amb}		0 to + 70 °C

D.C. CHARACTERISTICS

V_{SS} = 0 V; T_{amb} = 0 to + 70 °C; all voltages with respect to V_{SS}; unless otherwise specified

parameter	symbol	min.	typ.	max.	unit	conditions
Supply voltage	V_{CC} = V_{DD}	4,5	5,0	5,5	V	
Supply current total	I_{tot}	–	–	15	mA	f = 11 MHz; $I_{tot} = I_{CC} + I_{DD}$
		–	–	8,5	mA	f = 6 MHz; $I_{tot} = I_{CC} + I_{DD}$
		–	–	3	mA	f = 1 MHz; $I_{tot} = I_{CC} + I_{DD}$
IDLE mode	I_{idle}	–	–	6	mA	f = 11 MHz; $I_{idle} = I_{CC} + I_{DD}$
		–	–	4	mA	f = 6 MHz; $I_{idle} = I_{CC} + I_{DD}$
		–	–	1,2	mA	f = 1 MHz; $I_{idle} = I_{CC} + I_{DD}$
power down mode	I_{pd}	–	–	2	μA	V_{DD} = 2 V; $\overline{RESET}$ = LOW
Inputs						
Input voltage LOW all inputs except $\overline{RESET}$; XTAL 1; XTAL 2	V_{IL}	−0,5	–	0,18 x V_{CC}	V	
Input voltage LOW $\overline{RESET}$; XTAL 1; XTAL 2	V_{IL1}	−0,5	–	0,13 x V_{CC}	V	
Input voltage HIGH all inputs except $\overline{RESET}$; XTAL 1; XTAL 2	V_{IH}	0,4 x V_{CC}	–	V_{CC}	V	
Input voltage HIGH $\overline{RESET}$; XTAL 1; XTAL 2	V_{IH1}	0,7 x V_{CC}	–	V_{CC}	V	
Outputs						
Output voltage LOW BUS	V_{OL}	–	–	0,45	V	I_{OL} = 2 mA
Output voltage LOW $\overline{RD}$; $\overline{WR}$; $\overline{PSEN}$; ALE	V_{OL1}	–	–	0,45	V	I_{OL} = 1,8 mA
Output voltage LOW PROG	V_{OL2}	–	–	0,45	V	I_{OL} = 1 mA
Output voltage LOW all other outputs	V_{OL3}	–	–	0,45	V	I_{OL} = 1,6 mA

parameter	symbol	min.	typ.	max.	unit	conditions
Output voltage HIGH BUS	V_{OH}	0,75 x V_{CC}	–	–	V	$-I_{OH}$ = 400 μA
Output voltage HIGH $\overline{RD}$; $\overline{WR}$; $\overline{PSEN}$; ALE	V_{OH1}	0,75 x V_{CC}	–	–	V	$-I_{OH}$ = 100 μA
Output voltage HIGH all other outputs	V_{OH2}	0,75 x V_{CC}	–	–	V	$-I_{OH}$ = 40 μA
Input leakage current $\overline{INT}$; T1; EA	$\pm I_{IL}$	–	–	10	μA	without internal pull-up; $V_{SS} < V_I < V_{CC}$
Input leakage current P10 – P17; P20 – P27; $\overline{SS}$	$-I_{IL1}$	–	–	500	μA	with internal pull-up; $V_{SS} + 0{,}45 < V_I < V_{CC}$
Input leakage current $\overline{RESET}$	$-I_{ILR}$	20	–	300	μA	$V_{SS} < V_I \leqslant V_{IL1}$
Output leakage current BUS; T0 at high impedance state	$\pm I_{OL}$	–	–	10	μA	$V_{SS} + 0{,}45 < V_I < V_{CC}$

DEVELOPMENT DATA

A.C. CHARACTERISTICS

$V_{CC} = V_{DD} = 5\ V \pm 10\%$; $V_{SS} = 0\ V$; $T_{amb} = 0$ to $+70\ ^{o}C$; note 1

See waveforms Figs 13, 14, 15, 16 and 17

parameter	f (t_{CY})	symbol	11 MHz		unit
			min.	max.	
ALE pulse width	$7/30t_{CY}-170$	t_{LL}	150	–	ns
Address set-up time to ALE	$1/5t_{CY}-110$	t_{AL}	160	–	ns
Address hold time from ALE	$1/15t_{CY}-40$	t_{LA}	50	–	ns
Control pulse width $\overline{RD}$, $\overline{WR}$	$1/2t_{CY}-200$	t_{CC1}	480	–	ns
Control pulse width $\overline{PSEN}$	$2/5t_{CY}-200$	t_{CC2}	350	–	ns
Data set-up time before $\overline{WR}$	$13/30t_{CY}-200$	t_{DW}	390	–	ns
Data hold time after $\overline{WR}$ (note 2)	$1/15t_{CY}-50$	t_{WD}	40	–	ns
Data hold time $\overline{RD}$, $\overline{PSEN}$	$1/10t_{CY}-30$	t_{DR}	0	110	ns
$\overline{RD}$ to data input	$2/5t_{CY}-170$	t_{RD1}	–	350	ns
$\overline{PSEN}$ to data input	$3/10t_{CY}-170$	t_{RD2}	–	190	ns
Address set-up time to $\overline{WR}$	$1/3t_{CY}-150$	t_{AW}	300	–	ns
Address set-up time to data input ($\overline{RD}$)	$7/10t_{CY}-250$	t_{AD1}	–	730	ns
Address set-up time to data input ($\overline{PSEN}$)	$1/2t_{CY}-220$	t_{AD2}	–	460	ns
Address floating to $\overline{RD}$, $\overline{WR}$	$2/15t_{CY}-40$	t_{AFC1}	140	–	ns
Address floating to $\overline{PSEN}$	$1/30t_{CY}-40$	t_{AFC2}	10	–	ns
ALE to control pulse $\overline{RD}$, $\overline{WR}$	$1/5t_{CY}-75$	t_{LAFC1}	200	–	ns
ALE to control pulse $\overline{PSEN}$	$1/10t_{CY}-75$	t_{LAFC2}	60	–	ns
Control pulse to ALE $\overline{RD}$, $\overline{WR}$, $\overline{PROG}$	$1/15t_{CY}-40$	t_{CA1}	50	–	ns
Control pulse to ALE $\overline{PSEN}$	$4/15t_{CY}-40$	t_{CA2}	320	–	ns
Port control set-up to $\overline{PROG}$	$1/10t_{CY}-80$	t_{CP}	50	–	ns

parameter	f (t_{CY})	symbol	11 MHz		unit
			min.	max.	
Port control hold to $\overline{PROG}$	$4/15t_{CY}-260$	t_{PC}	100	–	ns
$\overline{PROG}$ to time P_2 input must be valid	$17/30t_{CY}-120$	t_{PR}	–	650	ns
Input data hold time from $\overline{PROG}$	$1/10t_{CY}$	t_{PF}	0	140	ns
Output data set-up time	$2/5t_{CY}-290$	t_{DP}	250	–	ns
Output data hold time	$1/10t_{CY}-90$	t_{PD}	40	–	ns
$\overline{PROG}$ pulse width	$7/10t_{CY}-250$	t_{PP}	700	–	ns
Port 2 I/O data set-up time to ALE	$4/15t_{CY}-200$	t_{PL}	160	–	ns
Port 2 I/O data hold time to ALE	$1/10t_{CY}-120$	t_{LP}	15	–	ns
Port output from ALE	$3/10t_{CY}+100$	t_{PV}	–	510	ns
Cycle time	$(1/f_{XTAL}) \times 15$	t_{CY}	1,36	15	μs
T0 repetition rate	$3/15t_{CY}$	t_{0PRR}	270	–	ns

Notes to A.C. characteristics

1. Control outputs: C_L = 80 pF
   Bus outputs: C_L = 150 pF.
2. Bus high-impedance load: 20 pF.
3. Interrupt pin must remain low for at least $3t_{CY}$ to ensure proper operation.

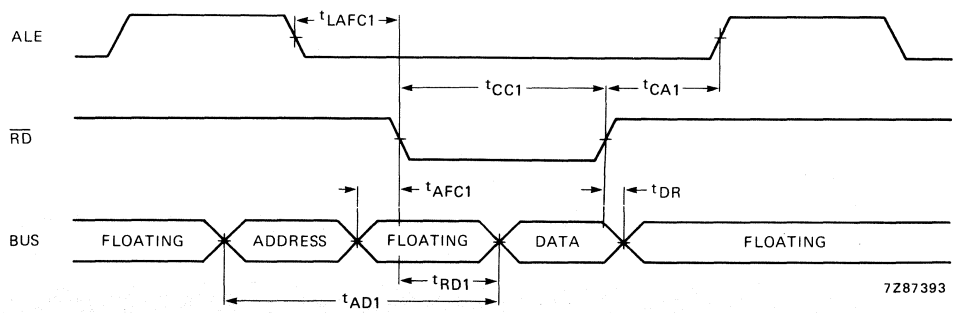


Fig. 13 Read from external data memory.

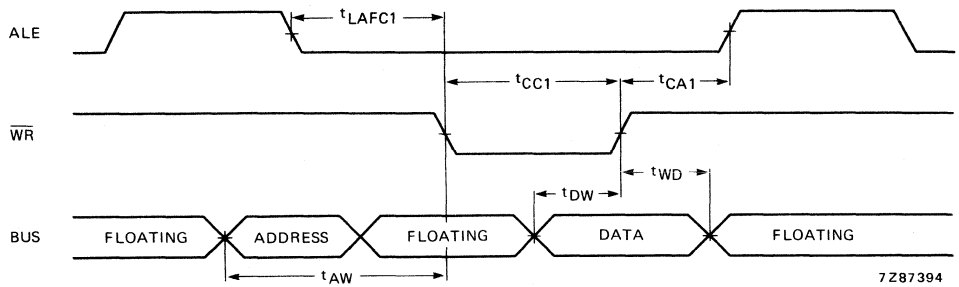


Fig. 14 Write to external data memory.

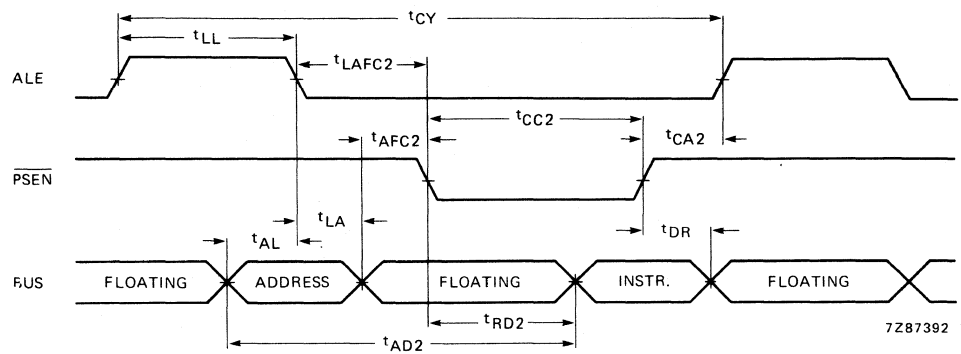


Fig. 15 Instruction fetch from external program memory.

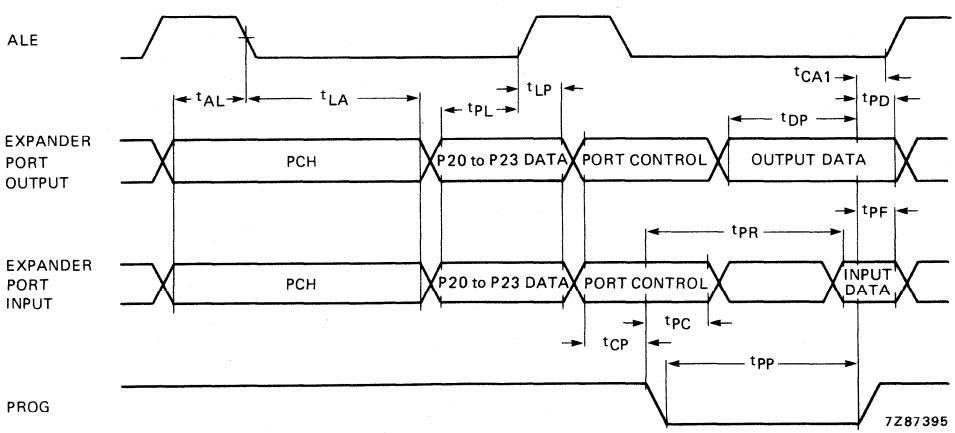


Fig. 16 Port 2 timing.

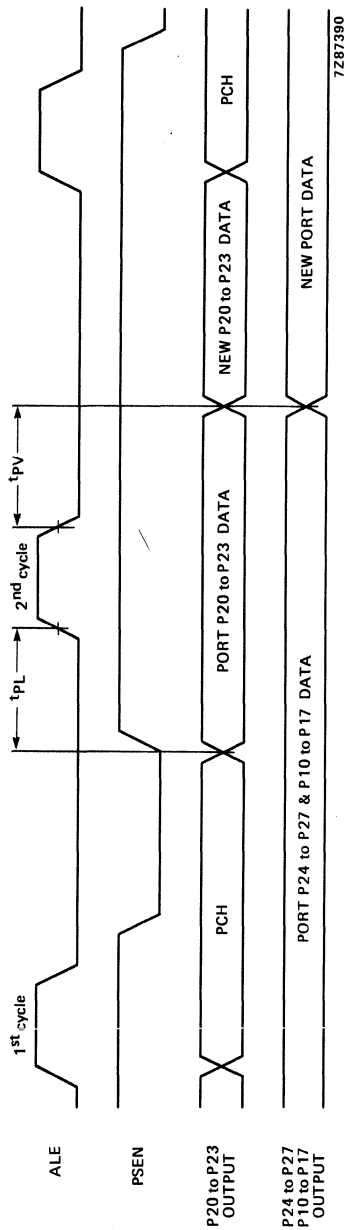


Fig. 17 I/O port timing.

CMOS MICROCONTROLLER FOR TELEPHONE SETS

GENERAL DESCRIPTION

The PCD3315C is a single-chip 8-bit microcontroller fabricated in CMOS and is a member of the PCD3343 family. It has special on-chip features for application in telephone sets.

Features

- 8-bit CPU, ROM, RAM, I/O in a single 28-lead DIL or SO package
- 1536 ROM bytes
- 160 RAM bytes
- 20 quasi-bidirectional I/O port lines
- Two test inputs: one of which is also the external interrupt input (CE/$\overline{T0}$)
- Single-level vectored interrupts: external, timer/event counter
- 8-bit programmable timer/event counter
- Over 80 instructions (based on MAB8048, MAB8400, PCD3343 and PCF8500)
- All instructions 1 or 2 cycles
- Clock frequency 100 kHz to 10 MHz
- Single supply voltage from 1,8 V to 6 V
- Low standby voltage and current
- STOP and IDLE mode
- On-chip oscillator with output drive capability for peripherals
- Configuration of all I/O port lines individually selected by mask: pull-up, open drain or push-pull
- Power-on-reset circuit and low supply voltage detection
- Reset state of all ports individually selected by mask
- Operating temperature range: −25 to + 70 °C

PACKAGE OUTLINES

PCD3315CP: 28-lead DIL; plastic (SOT-117D).
PCD3315CT: 28-lead mini-pack; plastic (SO-28; SOT-136A).

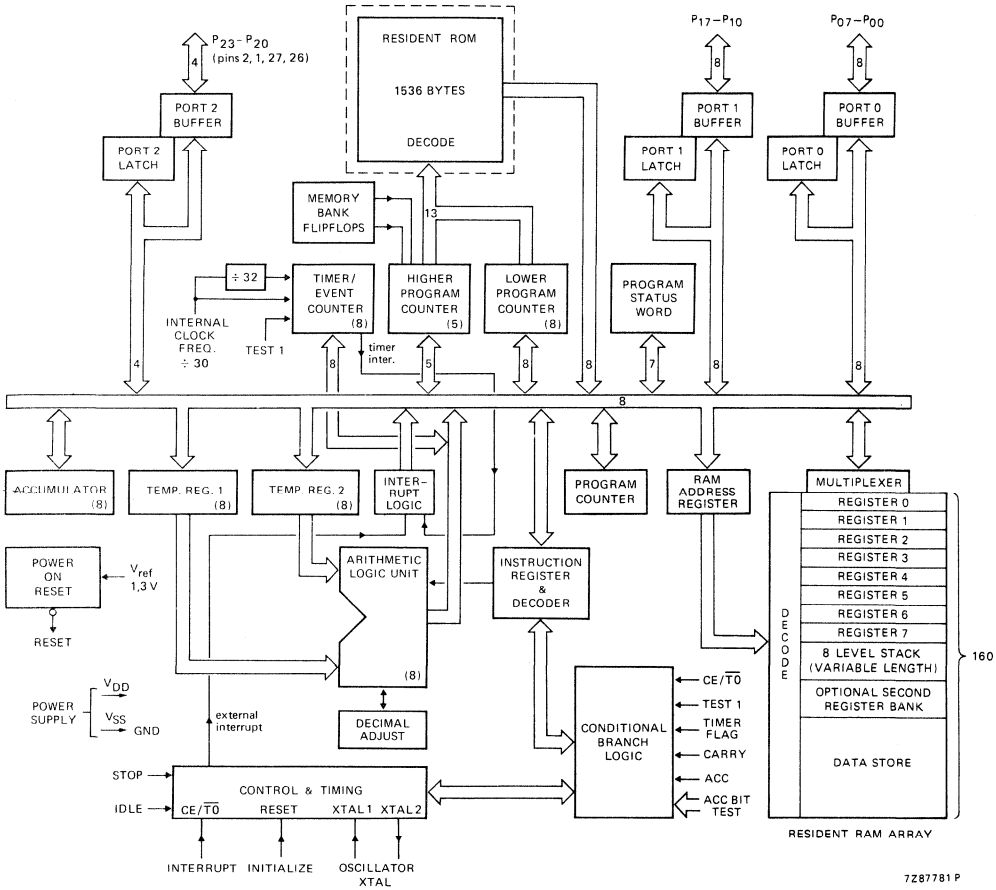
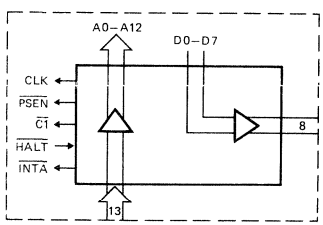
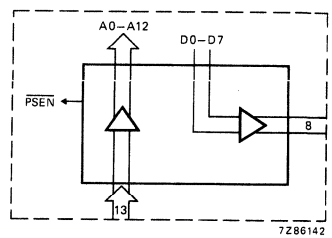


Fig. 1 Block diagram; PCD3315C.



(a)

Fig. 1a Replacement of dotted part in Fig. 1, for the PCD8500F bond-out version.



(b)

Fig. 1b Replacement of dotted part in Fig. 1, for the PCF8500B 'Piggy-back' version.

DEVELOPMENT DATA

PINNING

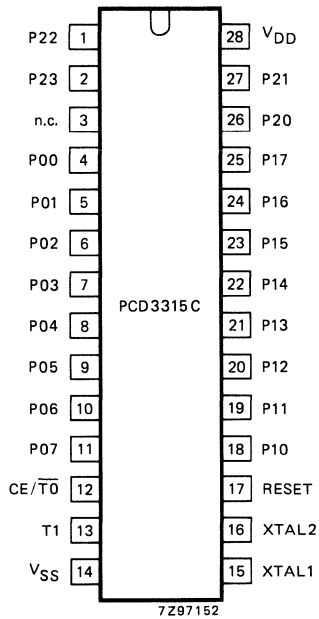


Fig. 2 Pinning diagram: PCD3315C.

PIN DESIGNATION

3	n.c.	not connected
4-11	P00-P07	**Port 0**: 8-bit quasi-bidirectional I/O port.
12	CE/$\overline{T0}$	**Interrupt/Test 0**: external interrupt input (sensitive to positive-going edge edge)/test input pin; when used as a test input directly tested by conditional branch instructions JT0 and JNT0.
13	T1	**Test 1**: test input pin, directly tested by conditional branch instructions JT1 and JNT1. T1 also functions as an input to the 8-bit timer/event counter, using the STRT CNT instruction.
14	V_{SS}	**Ground**: circuit earth potential.
15	XTAL 1	**Crystal input**: connection to timing component (crystal) which determines the frequency of the internal oscillator; also the input for an external clock source.
16	XTAL 2	connection to the other side of the timing component.
17	RESET	**Reset input**: used to initialize the processor (active HIGH), or output of power-on-reset circuit.
18-25	P10-P17	**Port 1**: 8-bit quasi-bidirectional I/O port.
26, 27, 1, 2	P20-P23	**Port 2**: 4-bit quasi-bidirectional I/O port.
28	V_{DD}	**Power supply**: 1,8 V to 6 V.

D.C. CHARACTERISTICS

V_{DD} = 2,5 to 6 V; V_{SS} = 0 V; T_{amb} = −25 to + 70 °C; all voltages with respect to V_{SS}; f = 3,58 MHz with R_S = 50 Ω; unless otherwise specified.

parameter	symbol	min.	typ.	max.	unit
Supply voltage					
operating	V_{DD}	1,8	–	6	V
STOP mode for RAM retention	V_{DD}	1,0	–	6	V
Supply current					
operating					
at V_{DD} = 3 V	I_{DD}	–	350	–	μA
IDLE mode					
at V_{DD} = 3 V	I_{DD}	–	150	–	μA
STOP mode (note 1)					
at V_{DD} = 1,8 V; T_{amb} = 25 °C	I_{DD}	–	1,2	2,5	μA
at V_{DD} = 1,8 V; T_{amb} = 55 °C	I_{DD}	–	–	5	μA
at V_{DD} = 1,8 V; T_{amb} = 70 °C	I_{DD}	–	–	10	μA
RESET I/O					
Switching level	V_{RESET}	–	1,2	–	V
Sink current					
at $V_{DD} > V_{RESET}$	I_{OL}	–	7	–	μA
Inputs					
Input voltage LOW	V_{IL}	0	–	$0{,}3V_{DD}$	V
Input voltage HIGH	V_{IH}	$0{,}7V_{DD}$	–	V_{DD}	V
Input leakage current					
at $V_{SS} < V_I < V_{DD}$	$\pm I_{IL}$	–	–	1	μA
Outputs					
Output voltage LOW					
at $V_I = V_{SS}$ or V_{DD}; $\lvert I_O \rvert < 1$ μA	V_{OL}	–	–	0,05	V
Output sink current LOW					
at V_{DD} = 3 V; V_O = 0,4 V	I_{OL}	0,6	1,5	–	mA
Pull-up output source current HIGH					
at V_{DD} = 3 V; $V_O = 0{,}9V_{DD}$	$-I_{OH}$	10	–	–	μA
at V_{DD} = 3 V; $V_O = V_{SS}$	$-I_{OH}$	–	–	200	μA
Push-pull output source current HIGH					
at V_{DD} = 3 V; $V_O = V_{DD}$−0,4 V	$-I_{OH}$	0,6	1,5	–	mA

Note 1

Crystal connected between XTAL 1 and XTAL 2; pin 2 pulled to V_{DD} via 5,6 kΩ resistor; CE and T1 at V_{SS}.

CMOS MICROCONTROLLER FOR TELEPHONE SETS

GENERAL DESCRIPTION

The PCD3343 is a single-chip 8-bit microcontroller fabricated in CMOS and is a member of the PCF8500 family. It has special on-chip features for application in telephone sets.
The device is mask programmable, designed to provide telephone dialling facilities such as redial, repertory dial, emergency call, keyboard scan and control for liquid crystal display, pulse dial and/or DTMF dial via dedicated peripheral.

Features

- 8-bit CPU, ROM, RAM, I/O in a single 28-lead DIL or SO package
- 3 K ROM bytes
- 224 RAM bytes
- 20 quasi-bidirectional I/O port lines
- Two test inputs: one of which is also the external interrupt input (CE/$\overline{T0}$)
- Single-level vectored interrupts: external, timer/event counter, serial I/O
- Serial I/O which can be used in bus systems with more than one master (serial I/O data via an existing port line and clock via a dedicated line)
- 8-bit programmable timer/event counter
- Over 80 instructions (based on MAB8048, MAB8400 and PCF8500)
- All instructions 1 or 2 cycles
- Clock frequency 100 kHz to 10 MHz
- Single supply voltage from 1,8 V to 6 V
- Low standby voltage and current
- STOP and IDLE mode
- On-chip oscillator with output drive capability for peripherals (e.g. PCD3312 DTMF generator)
- Configuration of all I/O port lines individually selected by mask: pull-up, open drain or push-pull
- Power-on-reset circuit and low supply voltage detection
- Reset state of all ports individually selected by mask
- Operating temperature range: −25 to + 70 °C

PACKAGE OUTLINES

PCD3343P : 28-lead DIL; plastic (SOT-117D).
PCD3343D: 28-lead DIL; ceramic (CERDIP) (SOT-135A).
PCD3343T: 28-lead mini-pack; plastic (SO-28; SOT-136A).

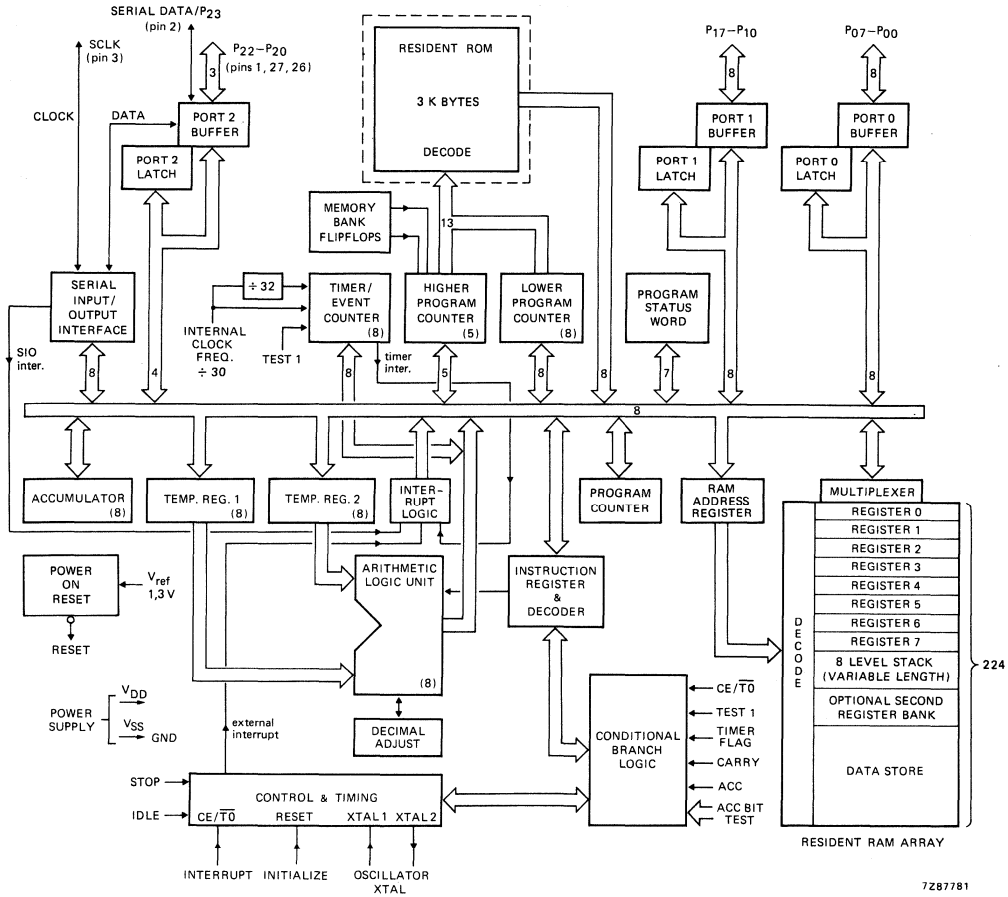
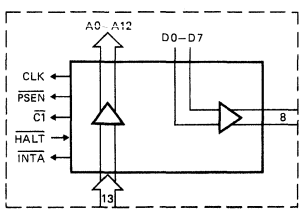
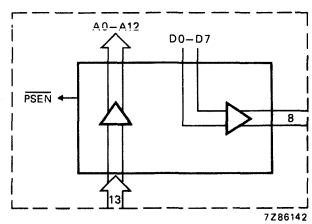


Fig. 1 Block diagram; PCD3343.



(a)

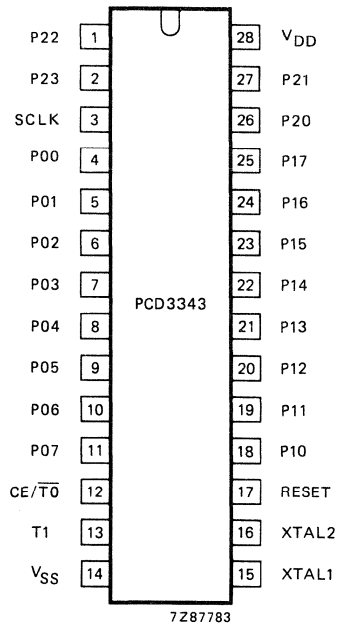
Fig. 1a Replacement of dotted part in Fig. 1, for the PCF8500F bond-out version.



(b)

Fig. 1b Replacement of dotted part in Fig. 1, for the PCF8500B 'Piggy-back' version.

PINNING



Note CE/$\overline{T0}$ is labelled $\overline{INT}$/T0 on the PCF8500B and has inverted polarity.

Fig. 2 Pinning diagram: PCD3343 and bottom pinning PCF8500B.

DEVELOPMENT DATA

PIN DESIGNATION

3	SCLK	**Clock**: bidirectional clock for serial I/O.
4-11	P00-P07	**Port 0**: 8-bit quasi-bidirectional I/O port.
12	CE/$\overline{T0}$	**Interrupt/Test 0**: external interrupt input (sensitive to positive-going edge)/test input pin; when used as a test input directly tested by conditional branch instructions JT0 and JNT0.
13	T1	**Test 1**: test input pin, directly tested by conditional branch instructions JT1 and JNT1. T1 also functions as an input to the 8-bit timer/event counter, using the STRT CNT instruction.
14	V_{SS}	**Ground**: circuit earth potential.
15	XTAL 1	**Crystal input**: connection to timing component (crystal) which determines the frequency of the internal oscillator; also the input for an external clock source.
16	XTAL 2	connection to the other side of the timing component.
17	RESET	**Reset input**: used to initialize the processor (active HIGH), or output of power-on-reset circuit.
18-25	P10-P17	**Port 1**: 8-bit quasi-bidirectional I/O port.
26, 27, 1, 2	P20-P23	**Port 2**: 4-bit quasi-bidirectional I/O port. P23 is the serial data input/output in serial I/O mode.
28	V_{DD}	**Power supply**: 1,8 V to 6 V.

PINNING (continued)

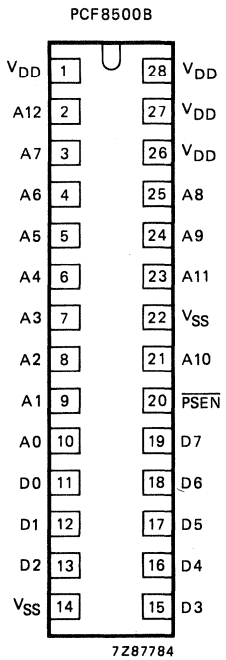


Fig. 3 Pinning diagram: PCF8500B 'Piggy-back' version top pinning; to access a 2732 or 2764 EPROM.

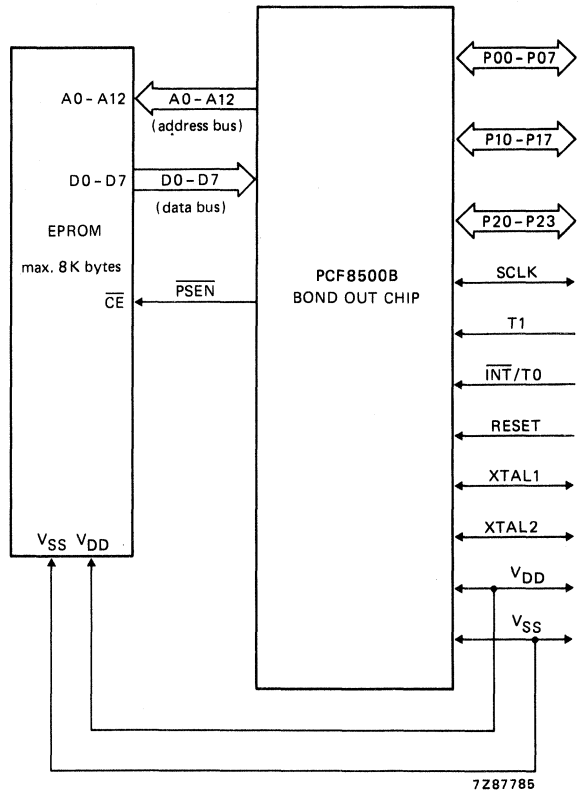


Fig. 3a Connection of EPROM to 'Piggy-back' package PCF8500B.

PIN DESIGNATION

14, 22	V_{SS}	**Ground**
1, 26-28	V_{DD}	**Power supply**
10-3, 25, 24, 21, 23, 2	A0-A12	**Address outputs**
11-13, 15-19	D0-D7	**Data**
20	$\overline{PSEN}$	**Program store enable**

Notes

1. RAM capacity of PCF8500B is 256 bytes.
2. Access time for ROMS/EPROMS to be below 7 x $1/f_{XTAL}$.
3. Pin 12 CE/$\overline{T0}$ is on the PCF8500B, inverted and labelled $\overline{INT}$/T0.

FUNCTIONAL DESCRIPTION

Bond-out version PCF8500F

The PCF8500F is a microcontroller that contains no on-board ROM, but has all address and data lines brought-out to access an external ROM or EPROM. This version has more pins than the PCD3343 with on-board ROM (see Fig. 1a). The RAM has 256 bytes. It can address 8 K bytes of ROM.

'Piggy-back' version PCF8500B

The PCF8500B is a special package that has standard pinning to the bottom which facilitates insertion as a mask-programmed device. An EPROM can be mounted on top in an additional socket. The total package height is greater than the standard DIL package. The RAM has 256 bytes and can also address 8 K bytes of program memory.

Program memory PCD3343

The program memory consists of 3072 bytes (8-bit words), in a read-only memory (ROM). Each location is directly addressable by the program counter. The memory is mask-programmed at the factory. Figure 4 shows the program memory map.

Four program memory locations are of special importance:

- Location 0; contains the first instruction to be executed after the processor is initialized (RESET),
- Location 3; contains the first byte of an external interrupt service subroutine,
- Location 5; contains the first byte of a serial I/O interrupt service subroutine,
- Location 7; contains the first byte of a timer/event counter interrupt service subroutine.

Program memory is arranged in banks of 2 K bytes, which are selected by SEL MB instructions. The program memory is further divided into location 'pages', each of 256 bytes. This latter division applies only for conditional branches. Memory bank boundaries can be crossed only by using the unconditional branch instructions after the appropriate memory bank has been selected. A CALL instruction can transfer control to a subroutine on any 'page'; RET and RETR instructions can transfer control from a subroutine back to the main program.

Data memory PCD3343

Data memory consists of 224 bytes (8-bit words), random-access data memory (RAM). All locations are indirectly addressable using RAM pointer registers; up to 16 designated locations are directly addressable. Memory also includes an 8-level program counter stack addressed by a 3-bit stack pointer. Figure 5 shows the data memory map.

Working registers

Locations 0 to 7 are designated as working registers, directly addressable by the direct register instructions. Ease of addressing, and a minimum requirement of instruction bytes to manipulate their contents, makes these locations suitable for storing frequently addressed intermediate results. This bank of registers can be selected by the SEL RB0 instruction.

Executing the select register bank instruction SEL RB1, designates locations 24 to 31 as working registers, instead of locations 0 to 7, and these are then directly addressable. This second bank of working registers may be used as an extension of the first or reserved for use during interrupt service subroutines saving the first bank for use in the main program. If the second bank is not used, locations 24 to 31 may serve as general purpose RAM.

The first locations of each bank contain the RAM pointer registers R0, R1, R0' and R1', which indirectly address all RAM locations.

All RAM locations make efficient program loop counters when used with the decrement register and test instruction DJNZ.

FUNCTIONAL DESCRIPTION (continued)

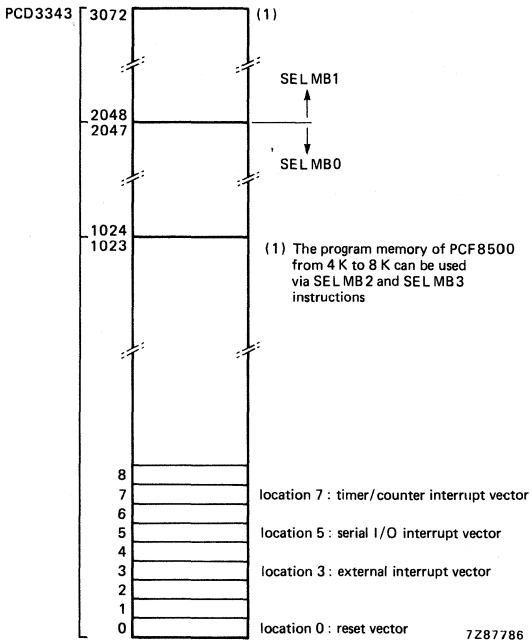


Fig. 4 Program memory map.

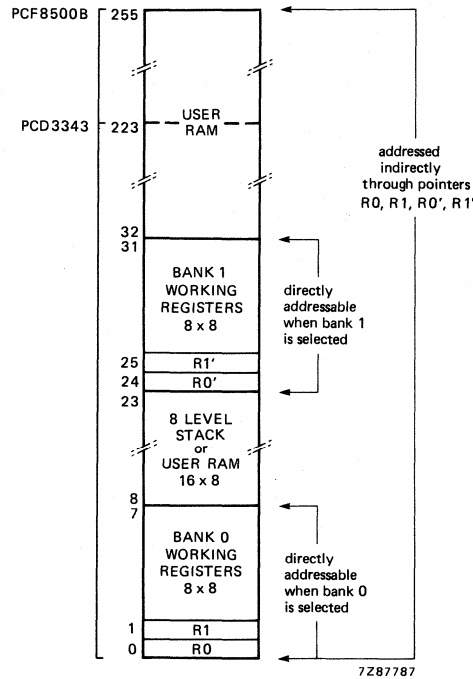


Fig. 5 Data memory map.

Program counter stack

Locations 8 to 23 may be designated as an 8-level program counter stack (2 locations per level), or as general purpose RAM. The program counter stack (Fig. 6) enables the processor to keep track of the return addresses and status generated by interrupts or CALL instructions by storing the contents of the program counter prior to servicing the subroutine. A 3-bit stack pointer determines which of the eight register pairs of the program counter stack will be loaded with next generated return address.

The stack pointer, when initialized to 000 by RESET, points to RAM locations 8 and 9. On the first subroutine CALL or interrupt, the contents of the program counter and bits 4, 6 and 7 of the program status word (PSW) are transferred to locations 8 and 9. The stack pointer increments by one and points to locations 10 and 11 ready for another CALL. Because an address may be up to 13 bits long, two bytes must be used to store each address.

At the end of a subroutine, which is signalled by a return instruction (RET or RETR), the stack pointer decrements by one and the contents of the register pair on top of the stack are transferred to the program counter. The saved PSW bits are transferred to the PSW only by the RETR instruction.

If not all 8 levels of subroutine and interrupt nesting are used, the unused portion of the stack may be used as any other indirectly addressable RAM locations. Possible locations from 32 to 223 may be used for storage of program variables or data.

Nesting of subroutines within subroutines can continue up to 8 times without overflowing the stack. If overflow does occur the deepest address stored (locations 8 and 9) will be overwritten and lost since the stack pointer overflows from 111 to 000. It also underflows from 000 to 111.

The value of the saved contents of the program counter is different for an interrupt CALL compared to a normal CALL to subroutine. With an interrupt CALL, the program counter return address is saved; with a subroutine CALL, the saved program counter value is one less than the program counter return address.

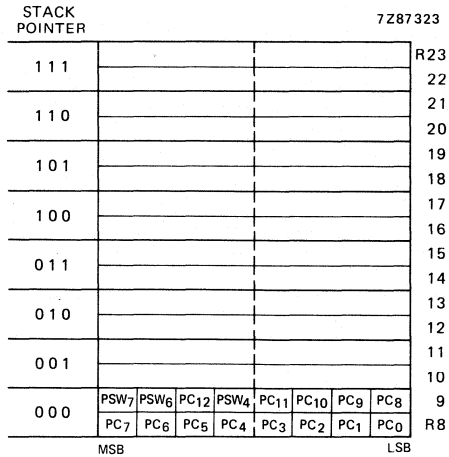


Fig. 6 Program counter stack.

IDLE and STOP modes

IDLE mode

When the microcontroller enters the IDLE mode via the IDLE instruction (H'01') the oscillator, timer/counter and serial I/O are kept running. The microcontroller exits from the IDLE mode by one of three interrupts if they are enabled or by activating a RESET. If the interrupt is not enabled the processor will remain in the IDLE mode. An active signal on the RESET pin restarts the microcontroller and a normal RESET sequence is executed (see Fig. 7).

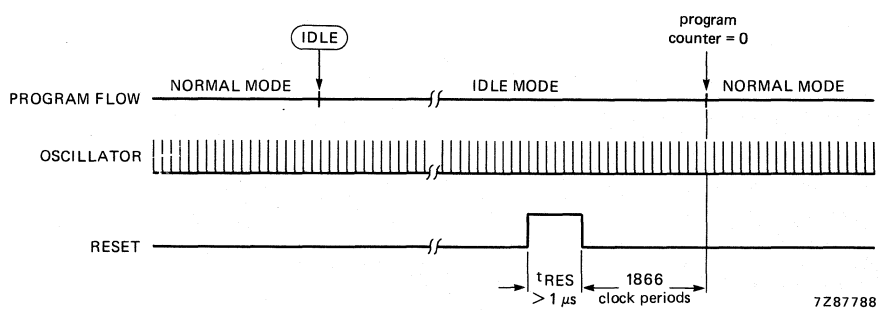


Fig. 7 Exit from IDLE mode via a RESET.

FUNCTIONAL DESCRIPTION (continued)

An active signal coming from an enabled interrupt causes the execution of the normal interrupt routine since normal interrupt scanning is still being carried out. A LOW-to-HIGH transition on the external interrupt pin (CE/$\overline{T0}$) reactivates the microcontroller. A HIGH level applied to CE/$\overline{T0}$ will reactivate the microcontroller only in the STOP mode. Thus, if CE/$\overline{T0}$ was HIGH before the microcontroller entered the IDLE mode, it must go LOW before the microcontroller can be reactivated (see Fig. 8).

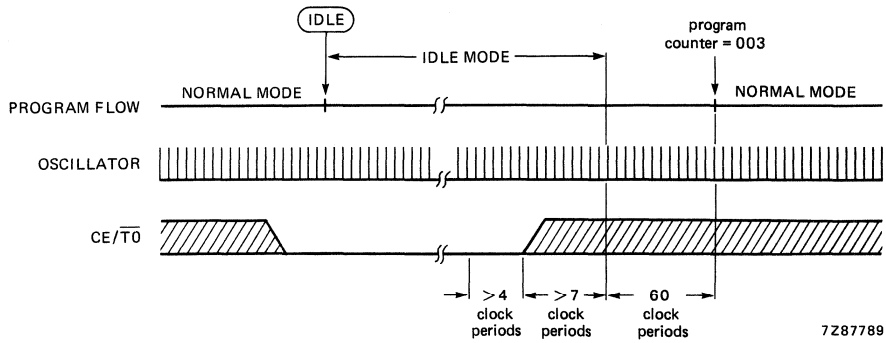


Fig. 8 Exit from IDLE mode via an interrupt.

Wake-up from the IDLE mode is ensured when CE/$\overline{T0}$ is LOW for 4 CP (clock periods) followed by a HIGH for 7 CP. After the initial forced CALL H'003' operation (60 CP) the program continues with the external interrupt service routine.

STOP mode

The microcontroller enters the STOP mode by the STOP instruction (H'22'). The oscillator is switched off. The internal status of the CPU, RAM contents and the state of I/O ports are not affected. The microcontroller can be brought-out of the STOP mode by an active signal at the external interrupt input or by an external RESET signal. When one of these two signals is applied an internal delay of 1866 CP is provided to ensure that all internal clocks are operating correctly before restart (see Fig. 9).

If the microcontroller exits from the STOP mode by activating RESET, a normal RESET sequence is executed.

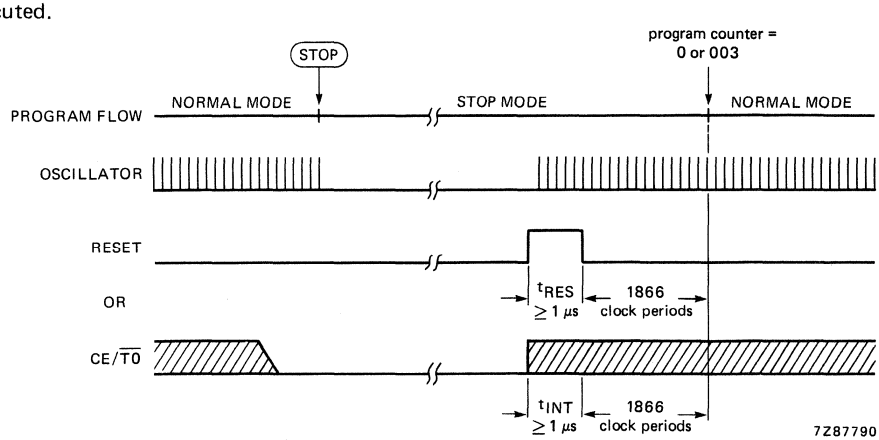


Fig. 9 Entering and exiting the STOP mode.

If the microcontroller exits the STOP mode by pulling the external interrupt input pin HIGH, an interrupt sequence is executed only if the external interrupt is enabled. In this event the microcontroller resumes the normal program sequence after returning from the interrupt routine, as in the normal mode. If the interrupt is not enabled, it continues the normal program sequence, executing the instruction following the STOP instruction.

The microcontroller is restarted by a HIGH level applied at the CE/$\overline{T0}$ pin, and not by a LOW-to-HIGH transition as in a normal interrupt mechanism.

When the CE/$\overline{T0}$ level is active during the STOP instruction then no STOP is executed.

A HIGH level on the external interrupt input of at least 1 μs will cause the microcontroller to exit the STOP mode.

I/O facilities

The PCD3343 family has 23 I/O lines arranged as:

- Port 0 parallel port of 8 lines (P00 to P07)
- Port 1 parallel port of 8 lines (P10 to P17)
- Port 2 parallel port of 4 lines (P20 to P23)
- SCLK serial I/O consisting of a data line shared with a parallel port line (P23) and a separate clock line SCLK
- CE/$\overline{T0}$ external interrupt and test input. When used as a test input can be directly tested by conditional branch instructions JT0 and JNT0
- T1 test input which can alter program sequences when tested by conditional jump instructions JT1 and JNT1. T1 also functions as an input to the 8-bit timer/event counter.

Parallel ports

All parallel ports can be used as outputs or inputs, their structure is quasi-bidirectional.
Output data written to a port is latched and remains unchanged until rewritten.
Input data is not latched and so must be present until read by an input instruction.

Input lines are fully CMOS compatible, output lines can drive one LS-TTL or CMOS load.

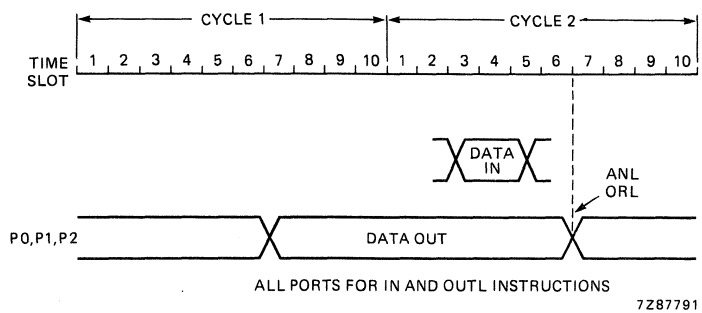


Fig. 10 Timing diagram of all ports on IN and OUTL instructions; for ANL and ORL instructions, the ports change on the time slot 7 of cycle 2.

Fig. 11 shows the quasi-bidirectional I/O interface with push-pull output and switched pull-up current source.
Each line is pulled up to V_{DD} via a constant current source (TR4), which is enabled via TR3 whenever one of the two output latches contains a logic 1. This current provides sufficient source current for a TTL HIGH level, yet can be pulled LOW by an external CMOS device, thus allowing the same pin to be used for both input and output.

FUNCTIONAL DESCRIPTION (continued)

When a logic 1 is written to the line for the first time (MQ = 1, SQ = 0), TR2 is switched on for the duration of the internal write pulse (one oscillator period), to provide a fast transition from logic 0 to logic 1. Subsequent writing of a logic 1 to the port lines will not switch TR2 on. This prevents unnecessary current through external components connected to the port lines of the same port which might be in the input mode and also connected to ground.

When a logic 0 is written to the line, TR3 switches off the current source. Current sinking capability is provided by TR1, which is now switched on. When used as an input, a logic 1 must first be written to the line, otherwise TR1 will remain low impedance.

In telephone applications this switched pull-up source may not be sufficient. Therefore the PCD3343 offers the possibility to select individually 19 of the 20 parallel port pins (not P23), by the following mask options:

Option 1- STANDARD PORT; quasi-bidirectional I/O with switched pull-up current source of 100 μA (typ.) and P-channel booster transistor TR2 (1,5 mA). TR2 is only active during 1 clock cycle (0,28 μs at 3,58 MHz).

Option 2- OPEN DRAIN; quasi-bidirectional I/O with only an N-channel open drain output. Application as an output requires connection of an external pull-up resistor (Fig. 12).

Option 3- PUSH-PULL OUTPUT; drive capability of the output will be 1,5 mA (typ.) at V_{DD} = 3 V in both polarities. To avoid a large current flowing through the output transistors during the input mode, these push-pull pins must only be used as outputs (Fig. 13).

Also, individual mask selection of the RESET state of these port pins can be achieved by appending the following options S and R to options 1, 2 or 3.

Option S-SET; after RESET this pin will be initialized to HIGH.

Option R-RESET; after RESET this pin will be initialized to LOW.

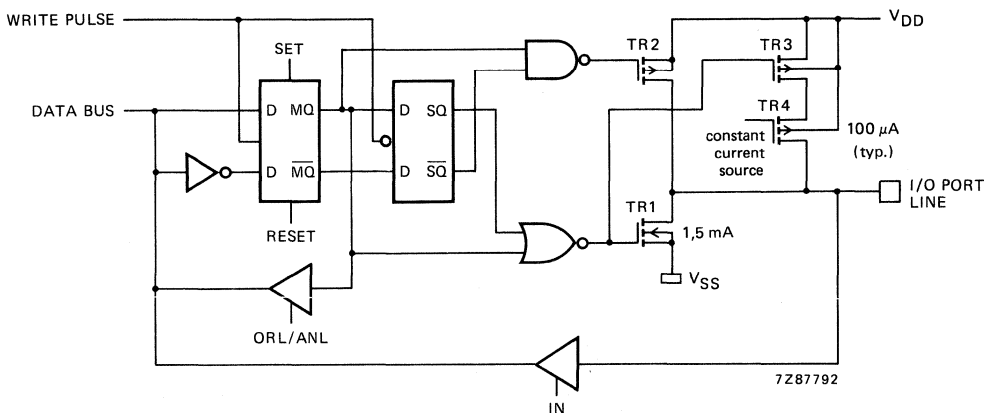


Fig. 11 Standard output with switched pull-up current source.

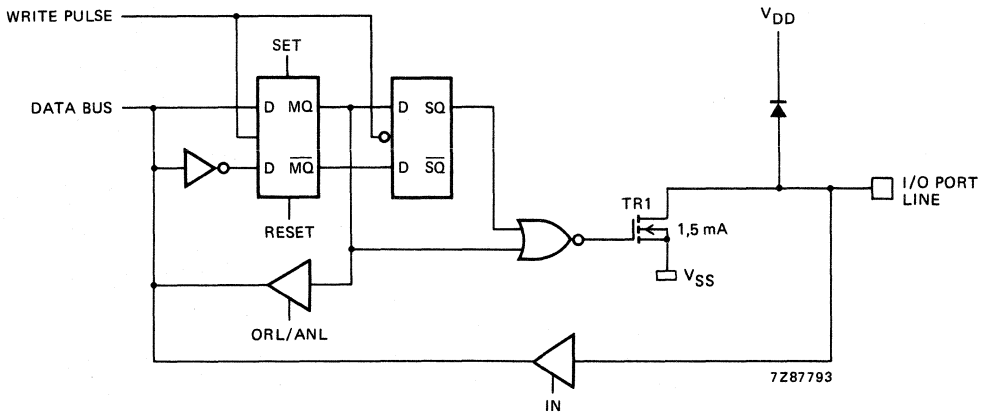


Fig. 12 Open drain output.

DEVELOPMENT DATA

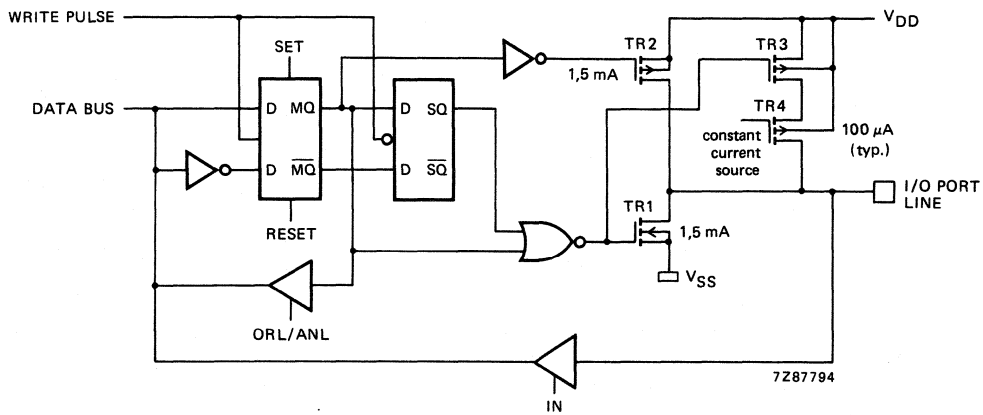


Fig. 13 Push-pull output.

FUNCTIONAL DESCRIPTION (continued)

Serial I/O (SIO)

The PCD3343 has an on-chip serial I/O interface. This SIO interface is a versatile feature in an intelligent telephone set, as shown in application diagram Fig. 32.

In this application the SIO is used to communicate with the different peripherals, such as:

- DTMF generator (PCD3312)
- LCD drivers (PCF8577)
- External RAM (PCD8571)
- Clock calendar (PCB8573)

No extra hardware is required for decoding, addressing and data processing.

Whereas a normal microcontroller must regularly monitor the serial data bus for the presence of data, the serial I/O interface detects, receives and converts the serial data stream into parallel format without interrupting the execution of the current program. An interrupt is sent to the PCD3343 only when a complete byte is received. It then reads the data byte in one instruction. Likewise during transmission the serial I/O interface performs parallel to serial conversion and subsequent serial output of the data. The microcontroller is only interrupted in the execution of its programmed tasks when a complete byte has been transmitted.

The design of the PCD3343 serial I/O system allows any number of devices from PCF8500 family (clips) to be connected via the two-line serial bus. The ability of any devices to communicate, without interrupting the operation of any other devices on the bus, is an outstanding attribute of the system. This is achieved by allocating a specific 7-bit address to each device and providing a system whereby a device reacts only to a message prefixed with its own address or the 'general CALL' address. Address recognition is performed by the interface hardware so that operation of the microcontroller need only be interrupted when a valid address has been received. This saves significant processing time and memory space compared with a conventional microcontroller employing a software serial interface. When the addressing facility is not required, for instance in a system with only two microcontrollers, direct data transfer without addressing can be performed. In multi-master systems, an automatically invoked arbitration procedure prevents two or more devices from continuing simultaneous transmission.

In NORMAL (running) and IDLE mode, the serial I/O logic remains active; its internal system clock will be switched off when there is no activity on the serial bus.

After execution of the STOP instruction, the oscillator of the PCD3343 is switched off. This means that the serial I/O logic will remain in the state it was at the occurance of the STOP instruction. To avoid "bus block" problems and to assure correct start-up of the bus after exit from the STOP mode, the user should disable the serial logic (ESO = 0) prior to the execution of the STOP instruction. This must be carried out only when the PCD3343 has finished a serial data transfer.

Serial I/O interface

Figure 14 shows the serial I/O interface. The clock line of the serial bus has exclusive use of pin 3 (SCLK) while the data line shares pin 2 (serial data) with the I/O line P23 of port 2. When the serial I/O is enabled, P23 is disabled as a parallel port line; (P23 and SCLK only open drain).

The microcontroller and interface communicate via the internal microcontroller bus and the Serial Interrupt Request line. Data and information controlling the operation of the interface are stored in four registers:

- Data shift register (S0)
- Serial I/O interface status word (S1)
- Serial clock control word (S2)
- Address register

Data shift register (S0)

Register S0 converts serial data to parallel format and vice versa. A pending interrupt is generated only after a complete byte has been transmitted, or after a complete data byte, specific address or 'general CALL' address has been received. The most significant bit is transmitted first.

Serial I/O interface status word (S1)

Register S1 provides information concerning the state of the interface and stores information from the microcontroller. Bits 0 to 3 are duplicated: control bits in these positions can only be written by the microcontroller, while interface bits can only be read.

MST and TRX (see Table 1)

These bits determine the operating mode of the serial I/O interface.

Table 1 Operating modes of the serial I/O interface

MST	TRX	operating mode
0	0	slave receiver
1	0	master receiver
0	1	slave transmitter
1	1	master transmitter

BB: Bus Busy.

This is the flag which indicates the status of the bus.

PIN: Pending Interrupt Not

PIN = '0' indicates the presence of a pending interrupt, which will cause a Serial Interrupt Request when the serial interrupt mechanism is enabled.

ESO: Enable Serial output

The ESO flag enables/disables the serial I/O interface: ESO = '1' enables, ESO = '0' disables. ESO can only be written by software.

BC0, BC1 and BC2

Bits BC0, BC1 and BC2 indicate the number of bits received or transmitted in a data stream. These bits can only be written by software.

AL: Arbitration Lost

The arbitration lost flag is set by hardware when the serial I/O interface, as master transmitter, loses a bus arbitration procedure.

AAS: Addressed As Slave

This flag is set by hardware when the interface detects either its own specific address or the 'general CALL' address as the first byte of a transfer and the interface has been programmed to operate in the address recognition mode.

AD0: Address Zero

This flag is set by hardware after detection of the 'general CALL' address when the interface is operating in the address recognition mode.

LRB: Last Received Bit

This contains either the last data bit received or, for a transmitting device in the acknowledgement mode, the acknowledgement signal from the receiving device.

Bits AL, AAS, AD0 and LRB can only be read by software.

FUNCTIONAL DESCRIPTION (continued)

Serial clock control word (S2)

Bits 0 to 4 of the clock control register S2 are used to set the frequency of the serial clock signal. When a 3,58 MHz crystal is used, the frequency of the serial clock can be varied between 92 kHz and 580 Hz (see Table 2). An asymmetrical clock with a HIGH-to-LOW ratio of 3 : 1 can be generated using bit 5. The asymmetrical clock allows a microcontroller more time per clock period for sampling the data line, making the timing of this action less critical. Bit 6 can be used to activate the acknowledge mode of the serial I/O. S2 is a write only register.

Address register

The address register contains the 7-bit address back-up latches and the bit (ALS) used to enable/disable the address recognition mode. The address register can be written using the MOV S0, A and MOV S0, # data instructions, but only when ES0 = '0'.

Serial I/O interrupt logic

An EN SI instruction enables and a DIS SI instruction disables the interrupt logic. When the logic is enabled, a pending interrupt results in a serial I/O interrupt to the processor, causing a CALL to location 5 in the ROM. When disabled, the presence of an interrupt is still indicated by PIN in S1, allowing the interrupt to be serviced. However, vectored interrupt will not occur.

DEVELOPMENT DATA

Table 2 SIO clock pulse frequency control when using a 3,58 MHz crystal

hexadecimal S20-S24 code	divisor	f_{SCLK} (kHz) (approximate)
0	not allowed	
1	39	92
2	45	80
3	51	70
4	63	57
5	75	48
6	87	41
7	99	36
8	123	29
9	147	24
A	171	21
B	195	18
C	243	15
D	291	12
E	339	11
F	387	9,2
10	483	7,4
11	579	6,2
12	675	5,3
13	771	4,6
14	963	3,7
15	1155	3,1
16	1347	2,7
17	1539	2,3
18	1923	1,9
19	2307	1,6
1A	2691	1,3
1B	3075	1,2
1C	3843	0,93
1D	4611	0,78
1E	5379	0,67
1F	6147	0,58

FUNCTIONAL DESCRIPTION (continued)

Table 3 Serial I/O addresses for telephony peripherals

type	address 7	6	5	4	3	2	1	0	description
PCF8570A	1	0	1	0	A2	A1	X	R/$\overline{W}$	2 K RAM
PCF8570	1	0	1	0	A2	A1	A0	R/$\overline{W}$	2 K RAM
PCD8571	1	0	1	0	A2	A1	A0	R/$\overline{W}$	1 K RAM
PCD3311	0	1	0	0	1	0	A0	R/$\overline{W}$	DTMF dialler
PCD3312	0	1	0	0	1	0	A0	R/$\overline{W}$	DTMF dialler
PCE2111	0	0	0	0	0	0	1	0	LCD driver *
PCD8573	1	1	0	1	0	A1	A0	R/$\overline{W}$	clock calendar
PCF8574	0	0	1	1	A2	A1	A0	R/$\overline{W}$	8-bit I/O expander
PCF8576	0	1	1	1	0	0	SA0	R/$\overline{W}$	1 : 4 LCD driver
PCF8577	0	1	1	1	0	1	0	R/$\overline{W}$	1 : 2 LCD driver

* LCD driver requires an additional enable line.

DEVELOPMENT DATA

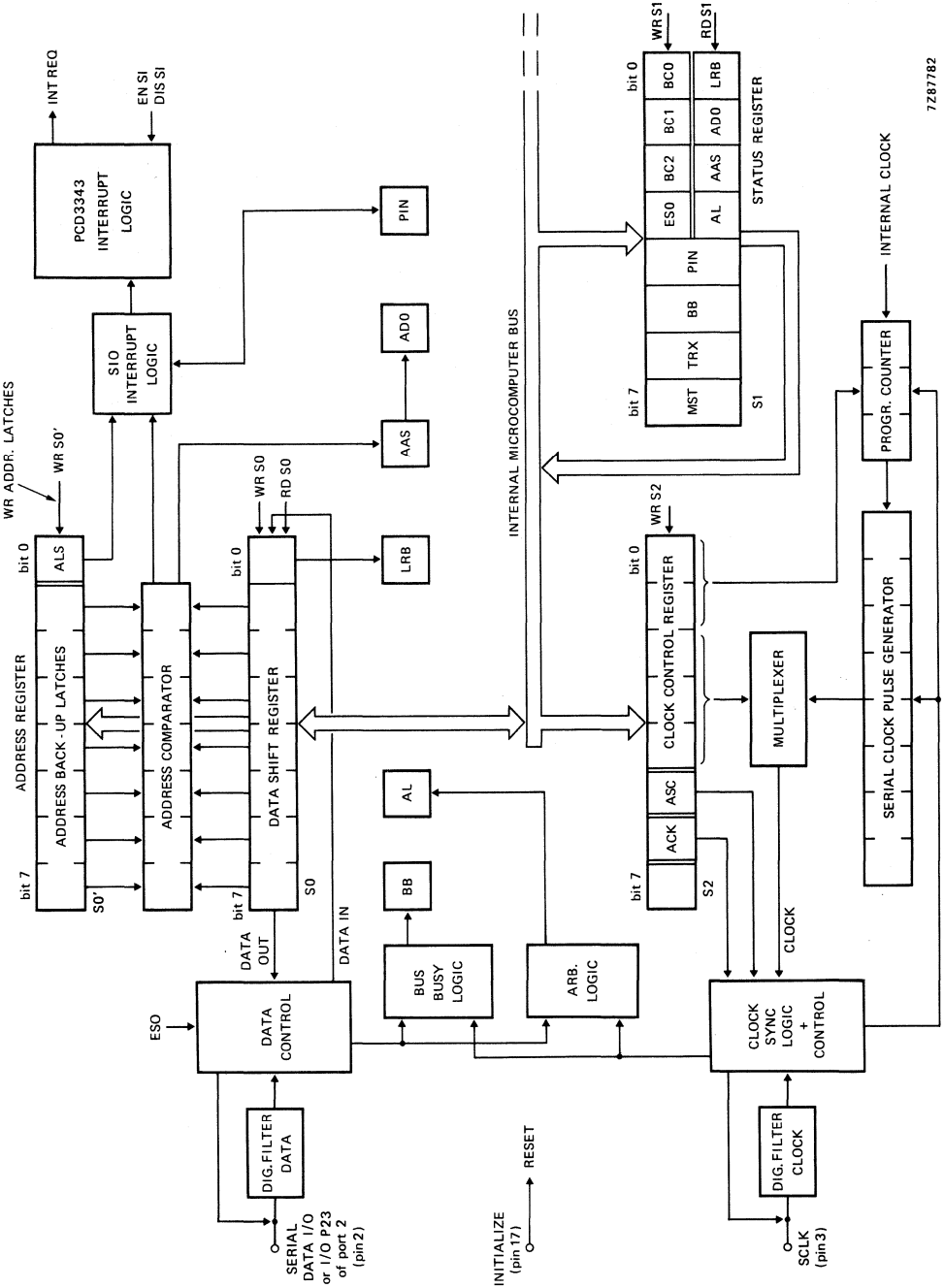


Fig. 14 Serial I/O interface.

FUNCTIONAL DESCRIPTION (continued)

Interrupts (see Fig. 15)

When the external interrupt is enabled, a LOW-to-HIGH transition on the CE/$\overline{T0}$ input initiates an external interrupt subroutine which causes a CALL to program memory location 3 following completion of the current instruction.
The interrupt must remain enabled until the interrupt instruction is completed, otherwise the next instruction of the main program will be executed. Serial I/O interrupt, when enabled, causes a CALL to location 5, and a timer/event counter overflow forces a CALL to location 7 when the timer interrupt is enabled.

When an interrupt subroutine starts, the contents of the program counter and bits 4, 6 and 7 of the PSW have been saved in the program counter stack. Accumulator contents have to be saved by software. Interrupt acknowledgement can be carried out by software via port pins. All interrupt subroutines must reside in memory bank 0.

Since the interrupt system is single level, once an interrupt is detected, all further interrupt requests are latched, but ignored, pending a RETR instruction to re-enable the interrupt input logic. After executing RETR, the program continues in the main part; this is independent of the occurance of a second interrupt during the running of the first routine. If 2 or 3 interrupts occur simultaneously, their priority is:

- (1) external
- (2) serial I/O
- (3) timer/event counter

An external interrupt can be generated by using the timer/counter in the event counter mode. The counter is first preset to (H'FF'), then EN TCNTI instruction is executed. A LOW-to-HIGH transition of the T1 input will then initiate an interrupt subroutine and cause a CALL to location 7.

On execution of a DIS I instruction, the PCD3343 always clears the digital filter/latch and the External Interrupt Flag.

The Timer Flag (TF) is reset only when the JTF or JNTF instruction is executed or after RESET.

The Timer Interrupt Flag is set when timer overflow occurs, only if the timer interrupt is enabled.

The microcontroller will exit the IDLE mode when any one of the following three interrupts is enabled:

- External
- Serial I/O
- Timer/event counter

There is no internal pull-up or pull-down device connected to the external interrupt input (pin 12). If required pin 12 must be externally connected to a resistor ($R \leqslant 100\ k\Omega$). When the external interrupt is not used pin 12 must be connected to V_{SS}.

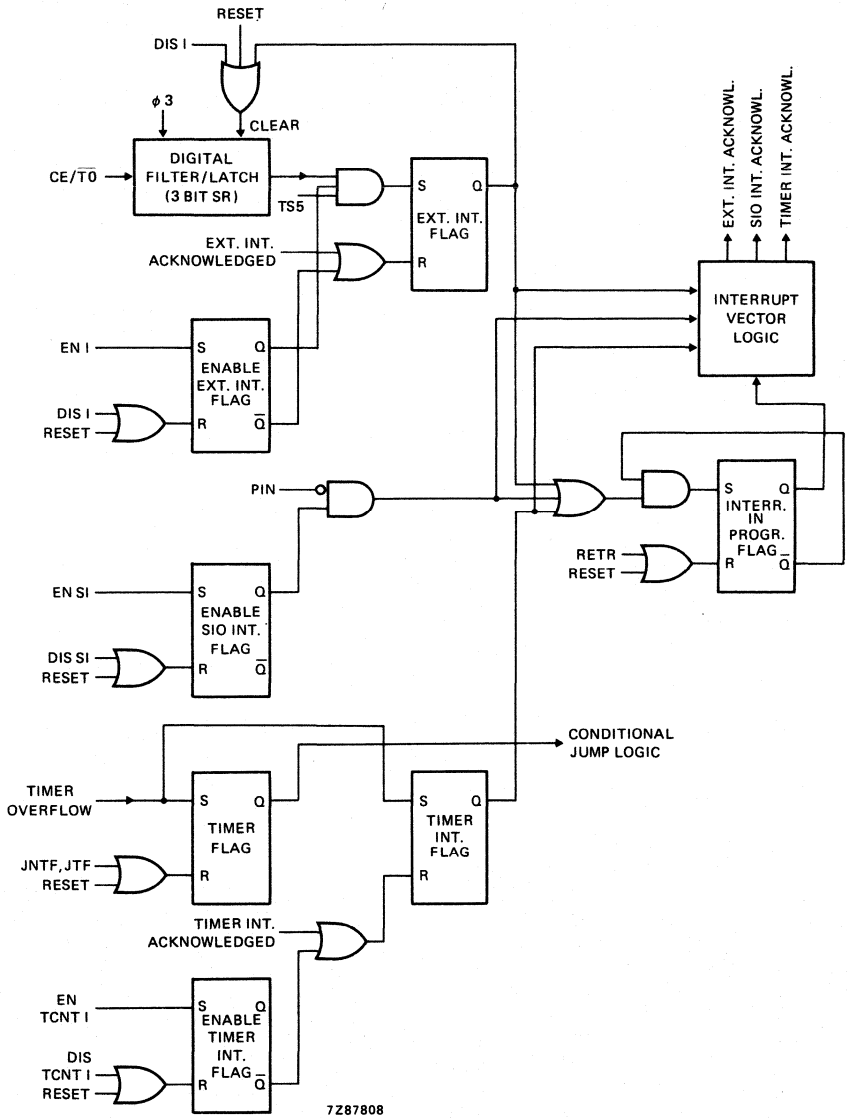


Fig. 15 Interrupt logic.

Notes to figure 15

1. CE/$\overline{T0}$ positive edge is always latched in the digital filter/latch.
2. Correct interrupt timing in ensured when CE/$\overline{T0}$ is LOW for >4 CP followed by a HIGH for >7 CP.
3. When the interrupt in progress flag is set, further interrupts are latched but ignored, until RETR is executed.
4. A DIS I instruction always clears a pending external interrupt.

FUNCTIONAL DESCRIPTION (continued)

Oscillator (see Fig. 16)

The 3,58 MHz oscillator can be inhibited by the STOP instruction under software control. It is also inhibited when a low-voltage condition is present to prevent discharge of a weak back-up battery.

Provided the supply voltage is within the operating range, the oscillator will be restarted after a STOP instruction by a HIGH level at either the CE/$\overline{T0}$ or RESET pin.

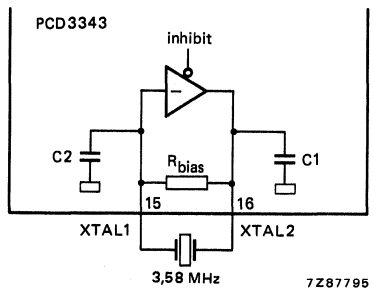


Fig. 16 Oscillator with integrated elements.

The oscillator has the output drive capability for the DTMF generator (PCD3311/3312) via pin 16 (XTAL 2). An external clock can be applied to pin 15 (XTAL 1). A machine cycle consists of 10 time slots, each time slot being 3 oscillator periods.

In telephony applications the 3,58 MHz crystal provides a 8,4 μs machine cycle. The range of the clock frequency is from 100 kHz up to a maximum which is a function of the supply voltage (see Fig. 23).

Timer/event counter (see Fig. 17)

An internal 8-bit up-counter is provided. This can count external events, modulo-32 machine cycles, or machine cycles directly. Table 4 gives the instructions that control the counter and the prescaler, and the functions performed.

When used as a timer, the input to the counter is either the overflow or input of a 5-bit prescaler. When used as an event counter, LOW-to-HIGH transitions on pin 13 (T1) are counted. The maximum rate at which the counter may be incremented is once every machine cycle (182,6 kHz for a 8,4 μs machine cycle). When the counter overflows, the timer flag is set. The flag can be tested and reset using the JTF (jump if timer flag = 1) or JNTF instruction. Overflow also generates an interrupt to the processor via setting of the Timer Interrpt Flag when the timer/event counter interrupt is enabled.

Table 4 Timer/event counter control

function	timer mode modulo-1, modulo-32*	counter mode
CLEAR	MOV T,A (A) = 0 or RESET	MOV T,A (A) = 0 or RESET
PRESET	MOV T,A	MOV T,A
START	STRT T	STRT CNT
STOP	STOP TCNT or RESET	STOP TCNT or RESET
TEST	JTF/JNTF	JTF/JNTF
READ**	MOV A,T	MOV A,T

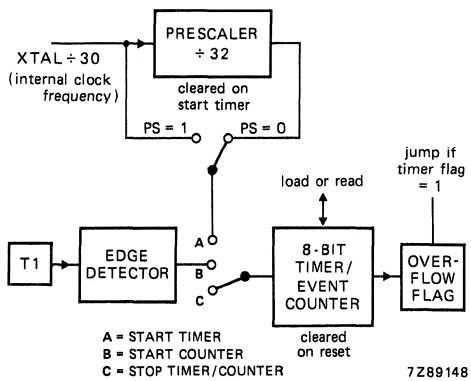


Fig. 17 Timer/event counter.

DEVELOPMENT DATA

Program status word (see Fig. 18)

The program status word (PSW) is an 8-bit word (1 byte) in the CPU which stores information about the current status of the microcontroller.
The PSW bits are:

- Bits 0 to 2 — stack pointer bits (SP_0, SP_1, SP_2)
- Bit 3 — prescaler select (PS);
  0 = modulo-32; 1 = modulo-1 (no prescaling)
- Bit 4 — working register bank select (RBS);
  0 = register bank 0; 1 = register bank 1
- Bit 5 — not used (1)
- Bit 6 — auxiliary carry (AC); half-carry bit generated by an ADD instruction and used by the decimal adjust instruction DA A
- Bit 7 — carry (CY); the carry flag indicates that previous operation has resulted in an overflow of the accumulator.

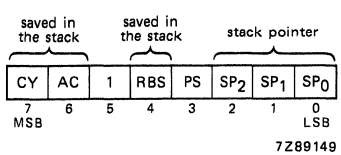


Fig. 18 Program status word.

* With prescaler select, PS = 0, the timer counts modulo-32 machine cycles, with PS = 1 it counts modulo-1 cycles (prescaler not used); prescaler cleared with STRT T, prescaler not readable.

** READ does not disturb the counting process.

FUNCTIONAL DESCRIPTION (continued)

Program status word (continued)

All bits can be read using the MOV A, PSW instruction. Bits 7 and 6 are set and cleared by CPU operation. Bit 4 can be changed by a SEL RB instruction, bit 3 by the MOV PSW, A instruction, and bits 0, 1 and 2 by the CALL, RET or RETR instructions and in the event of an interrupt. Bits 7, 6 and 4 are stored in the program counter stack during subroutine and interrupt calls. These bits are restored in the PSW with a RETR (return and restore) instruction which must be used at the end of an interrupt and can be used at the end of a normal subroutine. The RET instruction has no restore feature and cannot be used at the end of an interrupt.

Program counter (see Fig. 19)

A 13-bit program counter is used to facilitate 8 K bytes of ROM being addressed. The arrangement of the bits is shown in figure 19. During an interrupt subroutine PC_{11} and PC_{12} are forced to logic 0. All 13 bits are saved in the stack during CALL and interrupt routines.

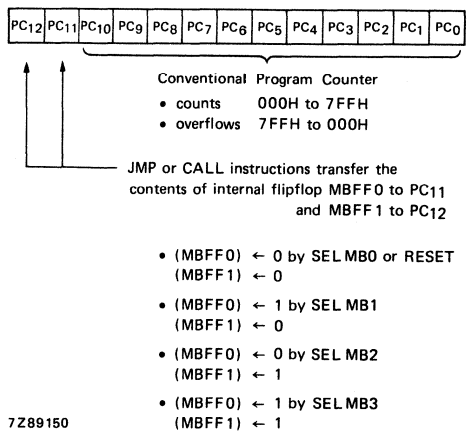


Fig. 19 Program counter.

Central processing unit

The PCD3343 has arithmetic, logical and branching capabilities. The DA A, SWAP A and XCHD instructions simplify BCD arithmetic and the handling of nibbles. The MOVP A,@A instruction permits efficient table look-up from the current ROM page.

Conditional branch logic

The conditional branch logic within the processor enables several conditions, internal and external to the processor, to be tested by the user's program. Table 5 lists the conditional jump instructions used to change the program sequence. The DJNZ instruction decrements a designated register or data memory location and branches if the contents are not zero. This instruction is useful for looping control. The JMPP@A instruction allows multiway branches to destinations determined by the contents of the accumulator.

Table 5 Conditional branches

test	jump condition	jump instruction
accumulator	all bits zero	JZ
	any bit non-zero	JNZ
accumulator bit test	1	JB0 to JB7
carry flag	1	JC
	0	JNC
timer overflow flag	1	JTF
	0	JNTF
test input T0	1	JNT0
	0	JT0*
test input T1	1	JT1
	0	JNT1
register	non-zero	DJNZ

Test input T1 (pin 13)

The T1 input line can be used as:

- A test input for branch instructions JT1 and JNT1
- An external input to the event counter

When used as a test input:

- JT1 instruction tests for logic 1 level
- JNT1 instruction tests for logic 0 level

When used as an input to the event counter, T1 must be LOW for > 4 CP, followed by a HIGH for > 4 CP. The transition can be recognized with a repetition rate of once per 30 oscillator clock periods (1 machine cycle).

There is no internal pull-up or pull-down resistor connected to the T1 input. If required it must be externally connected to a resistor ($R = \leqslant 100\ k\Omega$). When T1 is not used pin 13 must be connected to V_{DD} or V_{SS}.

Reset (pin 17)

A positive-going signal on the RESET input/output:

- Sets the program counter to zero
- Selects location 0 of memory band 0 and register bank 0
- Sets the stack pointer to zero (000); pointing to RAM address 8
- Disables the interrupts (external, timer and serial I/O)
- Stops the timer/event counter, then sets it to zero
- Sets the timer prescaler to modulo-32
- Resets the timer flag
- Sets all ports according to reset states
- Sets the serial I/O to slave receiver mode and disables the serial I/O
- Cancels IDLE and STOP mode

After the voltage is applied to RESET an internal delay of 1866 CP is introduced before the microcontroller commences operation.

* Because of the inverted interrupt input CE/$\overline{T0}$ the conditional jump JT0 is also inverted.

FUNCTIONAL DESCRIPTION (continued)

Power-on-reset and low-voltage detection (see Fig. 20)

In telephony applications, correct operation of the PCD3343 during moments of slowly changing supply voltages and low-voltage conditions is essential. This is achieved by the addition of an internal power-on-reset and low-voltage detection circuit.
To allow an external RESET signal being fed into the PCD3343, the reset pin (pin 17) has been configured as an input/output.

While a reset condition exists in the detection circuit, pin 17 is pulled HIGH by TR1 controlled by the reset circuit.
When the reset condition is not present a pull-down current source (TR2) will be activated. TR2 forces pin 17 LOW thus removing the RESET signal from the microcontroller.

Since the level at pin 17 is recognized by the microcontroller, the reset time constant can be stretched by connecting an external capacitor between V_{DD} and pin 17 (see Fig. 22).

The signal at pin 17 can also be used as an output to reset other devices in the system.

The internal reset circuit monitors the PCD3343 supply voltage. If the voltage drops below the switching level (typ. 1,3 V), a reset (HIGH) is applied to pin 17. This reset is removed (pin 17 goes LOW), after a fixed delay (t_d), when the supply voltage rises above the switching level again. The delay ensures a complete reset even when the supply voltage quickly rises above switching level after initial switch-on.

During a low-voltage condition the oscillator is inhibited to prevent complete discharge of a weak battery. The timing of the power-on-reset and low-voltage detection circuit is shown in figure 21.

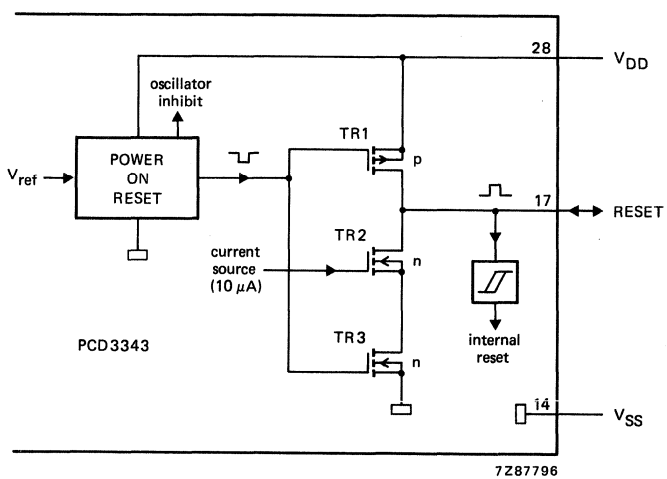
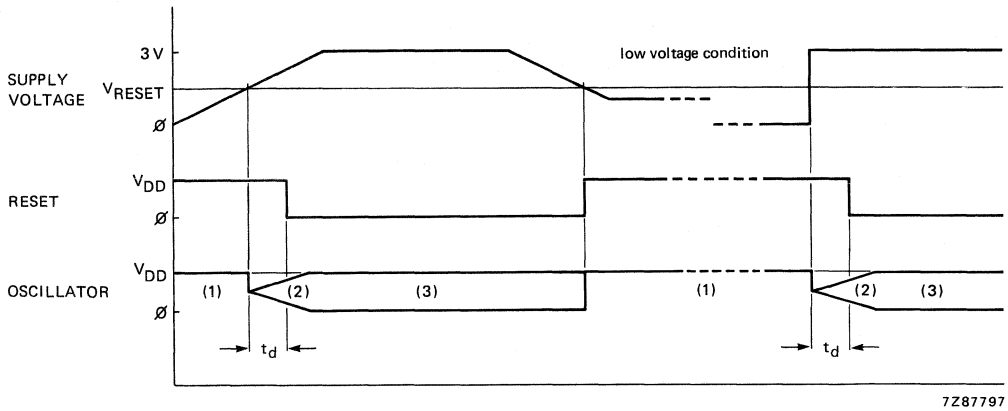


Fig. 20 Power-on-reset configuration.



Where: (1) Oscillator inhibited
(2) Oscillator starting
(3) Oscillator running, but may be stopped with a STOP condition

Fig. 21 Timing of power-on-reset and low-voltage detection.

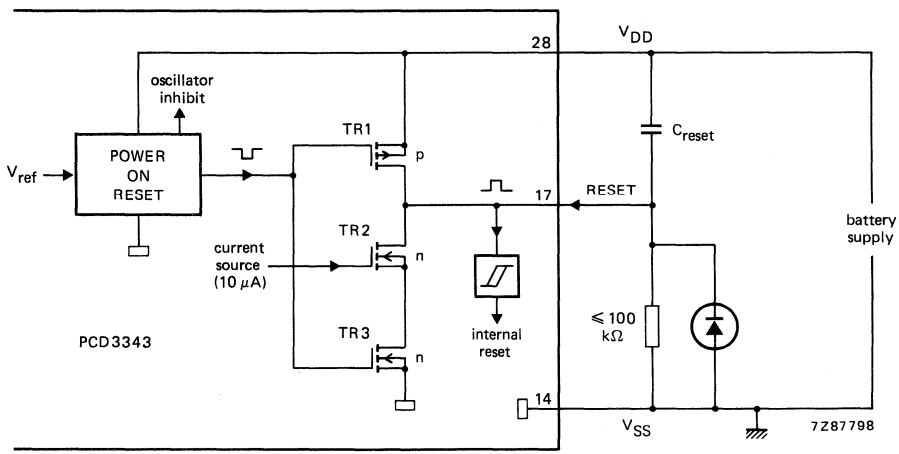


Fig. 22 Stretched power-on-reset with external capacitor.

INSTRUCTION SET

The PCD3343 instruction set consists of over 80 one and two byte instructions and is based on the MAB8048 instruction set. New instructions include those for serial I/O operation and memory bank selection. Program code efficiency is high because all RAM locations and all ROM locations on a 256 byte page require only a single byte address.

Table 8 gives the instruction set of the PCD3343. Table 7 shows the instruction map and Table 6 details the symbols and definition descriptions that are used.

Table 6 Symbols and definitions used in Table 8

symbol	definition description
A	accumulator
addr	program memory address
Bb	bit designation (b = 0-7)
RBS	register bank select
C	carry bit (bit CY)
CNT	event counter
D	mnemonic for 4-bit digit (nibble)
data	8-bit number or expression
I	interrupt
MB	memory bank
MBFF	memory bank flip-flop
P	mnemonic for 'in-page' operation
PC	program counter
Pp	port designation (p = 0, 1 or 2)
PSW	program status word
RB	register bank
Rr	register designation (r = 0-7)
Sn	serial I/O register
SP	stack pointer
T	timer
TF	timer flag
T0, T1	test 0 and 1 inputs
#	immediate data prefix
@	indirect address prefix
(X)	contents of X
((X))	contents of location addressed by X
←	is replaced by
↔	is exchanged with

Table 7 PCD3343 instruction map

first hexadecimal character of opcode ↓

second hexadecimal character of opcode →

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP	IDLE		ADD A,#data	JMP page 0	EN I	JNTF addr	DEC A	IN A,Pp				MOV A,Sn			
									0	1	2		0	1		
1	INC @Rr		JB0 addr	ADDC A,#data	CALL page 0	DIS I	JTF addr	INC A	INC Rr							
	0	1							0	1	2	3	4	5	6	7
2	XCH A,@Rr		STOP	MOV A,#data	JMP page 1	EN TCNTI	JNTO addr	CLR A	XCH A,Rr							
	0	1							0	1	2	3	4	5	6	7
3	XCHD A,@Rr		JB1 addr		CALL page 1	DIS TCNTI	JTO addr	CPL A	OUTL Pp,A				MOV Sn,A			
	0	1							0	1	2		0	1	2	
4	ORL A,@Rr		MOV A,T	ORL A,#data	JMP page 2	STRT CNT	JNT1 addr	SWAP A	ORL A,Rr							
	0	1							0	1	2	3	4	5	6	7
5	ANL A,@Rr		JB2 addr	ANL A,#data	CALL page 2	STRT T	JT1 addr	DA A	ANL A,Rr							
	0	1							0	1	2	3	4	5	6	7
6	ADD A,@Rr		MOV T,A		JMP page 3	STOP TCNT		RRC A	ADD A,Rr							
	0	1							0	1	2	3	4	5	6	7
7	ADDC A,@Rr		JB3 addr		CALL page 3			RR A	ADDC A,Rr							
	0	1							0	1	2	3	4	5	6	7
8				RET	JMP page 4	EN SI			ORL Pp,#data							
									0	1	2					
9			JB4 addr	RETR	CALL page 4	DIS SI	JNZ addr	CLR C	ANL Pp,#data				MOV Sn,#data			
									0	1	2		0	1	2	
A	MOV @Rr,A			MOVP A,@A	JMP page 5	SEL MB2		CPL C	MOV Rr,A							
	0	1							0	1	2	3	4	5	6	7
B	MOV @Rr,#data		JB5 addr	JMPP @A	CALL page 5	SEL MB3			MOV Rr,#data							
	0	1							0	1	2	3	4	5	6	7
C	DEC @Rr				JMP page 6	SEL RB0	JZ addr	MOV A,PSW	DEC Rr							
	0	1							0	1	2	3	4	5	6	7
D	XRL A,@Rr		JB6 addr	XRL A,#data	CALL page 6	SEL RB1		MOV PSW,A	XRL A,Rr							
	0	1							0	1	2	3	4	5	6	7
E	DJNZ @Rr,addr				JMP page 7	SEL MB0	JNC addr	RL A	DJNZ Rr, addr							
	0	1							0	1	2	3	4	5	6	7
F	MOV A,@Rr		JB7 addr		CALL page 7	SEL MB1	JC addr	RLC A	MOV A,Rr							
	0	1							0	1	2	3	4	5	6	7

INSTRUCTION SET (continued)

Table 8 Instruction set

	mnemonic	opcode (hex.)	bytes/ cycles	description	function		notes
ACCUMULATOR	ADD A, Rr	6*	1/1	Add register contents to A	(A)←(A) + (Rr)	r = 0–7	1
	ADD A, @Rr	60 61	1/1	Add RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) (A)←(A) + ((R1))		1
	ADD A, #data	03 data	2/2	Add immediate data to A	(A)←(A) + data		1
	ADDC A, Rr	7*	1/1	Add carry and register contents to A	(A)←(A) + (Rr) + (C)	r = 0–7	1
	ADDC A, @Rr	70 71	1/1	Add carry and RAM data, addressed by Rr, to A	(A)←(A) + ((R0)) + (C) (A)←(A) + ((R1)) + (C)		1
	ADDC A, #data	13 data	2/2	Add carry and immediate data to A	(A)←(A) + data + (C)		1
	ANL A, Rr	5*	1/1	'AND' Rr with A	(A)←(A) AND (Rr)	r = 0–7	
	ANL A, @Rr	50 51	1/1	'AND' RAM data, addressed by Rr, with A	(A)←(A) AND ((R0)) (A)←(A) AND ((R1))		
	ANL A, #data	53 data	2/2	'AND' immediate data with A	(A)←(A) AND data		
	ORL A, Rr	4*	1/1	'OR' Rr with A	(A)←(A) OR (Rr)	r = 0–7	
	ORL A, @Rr	40 41	1/1	'OR' RAM data, addressed by Rr, with A	(A)←(A) OR ((R0)) (A)←(A) OR ((R1))		
	ORL A, #data	43 data	2/2	'OR' immediate data with A	(A)←(A) OR data		
	XRL A, Rr	D*	1/1	'XOR' Rr with A	(A)←(A) XOR (Rr)	r = 0–7	
	XRL A, @Rr	D0 D1	1/1	'XOR' RAM, addressed by Rr, with A	(A)←(A) XOR ((R0)) (A)←(A) XOR ((R1))		
	XRL A, #data	D3 data	2/2	'XOR' immediate data with A	(A)←(A) XOR data		
	INC A	17	1/1	increment A by 1	(A)←(A) + 1		
	DEC A	07	1/1	decrement A by 1	(A)←(A) −1		
	CLR A	27	1/1	clear A to zero	(A)←0		
	CPL A	37	1/1	one's complement A	(A)←NOT(A)		
	RL A	E7	1/1	rotate A left	$(A_{n+1})\leftarrow(A_n)$ $(A_0)\leftarrow(A_7)$	n = 0–6	

DEVELOPMENT DATA

ACCUMULATOR (cont.)	RLC A	F7	1/1	rotate A left through carry	$(A_{n+1})\leftarrow A_n$ $(A_0)\leftarrow(C)$, $(C)\leftarrow(A_7)$	n = 0–6	2
	RR A	77	1/1	rotate A right	$(A_n)\leftarrow(A_{n+1})$ $(A_7)\leftarrow(A_0)$	n = 0–6	
	RRC A	67	1/1	rotate A right through carry	$(A_n)\leftarrow(A_{n+1})$ $(A_7)\leftarrow(C)$, $(C)\leftarrow(A_0)$	n = 0–6	2
	DA A	57	1/1	decimal adjust A			2
	SWAP A	47	1/1	swap nibbles of A	$(A_{4-7})\leftrightarrow(A_{0-3})$		
DATA MOVES	MOV A, Rr	F*	1/1	move register contents to A	(A)←(Rr)	r = 0–7	
	MOV A, @Rr	F0 F1	1/1	move RAM data, addressed by Rr, to A	(A)←((R0)) (A)←((R1))		
	MOV A, #data	23 data	2/2	move immediate data to A	(A)←data		
	MOV Rr, A	A*	1/1	move accumulator contents to register	(Rr)←(A)	r = 0–7	
	MOV @Rr, A	A0 A1	1/1	move accumulator contents to RAM location addressed by Rr	((R0))←(A) ((R1))←(A)		
	MOV Rr, #data	B* data	2/2	move immediate data to Rr	(Rr)←data		
	MOV @Rr, #data	B0 data B1 data	2/2	move immediate data to RAM location addressed by Rr	((R0))←data ((R1))←data		
	XCH A, Rr	2*	1/1	exchange accumulator contents with Rr	(A)↔(Rr)	r = 0–7	
	XCH A, @Rr	20 21	1/1	exchange accumulator contents with RAM data addressed by Rr	(A)↔((R0)) (A)↔((R1))		
	XCHD A, @Rr	30 31	1/1	exchange lower nibbles of A and RAM data addressed by Rr	$(A_{0-3})\leftrightarrow((R0_{0-3}))$ $(A_{0-3})\leftrightarrow((R1_{0-3}))$		
	MOV A, PSW	C7	1/1	move PSW contents to accumulator	(A)←(PSW)		
	MOV PSW, A	D7	1/1	move accumulator bit 3 to PSW_3	$(PSW_3)\leftarrow(A_3)$		3
	MOVP A, @A	A3	1/2	move indirectly addressed data in current page to A	$(PC_{0-7})\leftarrow(A)$, $(A)\leftarrow((PC))$		
FLAGS	CLR C	97	1/1	clear carry bit	(C)←0		2
	CPL C	A7	1/1	complement carry bit	(C)←NOT(C)		2

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
REGISTER	INC Rr	1*	1/1	increment register by 1	(Rr)←(Rr) + 1 r = 0–7	
	INC @Rr	10 11	1/1	increment RAM data, addressed by Rr, by 1	((R0))←((R0)) + 1 ((R1))←((R1)) + 1	
	DEC Rr	C*	1/1	decrement register by 1	(Rr)←(Rr) – 1 r = 0–7	
	DEC @Rr	C0 C1	1/1	decrement RAM data, addressed by Rr, by 1	((R0))←((R0)) – 1 ((R1))←((R1)) – 1	
BRANCH	JMP addr	● 4 address	2/2	unconditional jump within a 2 K bank	(PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1	
	JMPP @A	B3	1/2	indirect jump within a page	(PC_{0-7})←((A))	
	DJNZ Rr, addr	E* address	2/2	decrement Rr by 1 and jump if not zero to addr	(Rr)←(Rr) – 1 r = 0–7 if (Rr) not zero (PC_{0-7})←addr	
	DJNZ @Rr, addr	E0	2/2	decrement RAM data, addressed by Rr by 1 and jump if not zero to addr	((R0))←((R0)) – 1 if ((R0)) not zero (PC_{0-7})←addr	
		E1			((R1))←((R1)) – 1 if ((R1)) not zero (PC_{0-7})←addr	
	JBb addr	▲ 2 address	2/2	jump to addr if Acc. bit b = 1	if b = 1 : (PC_{0-7})←addr b = 0–7	
	JC addr	F6 address	2/2	jump to addr if C = 1	if C = 1 : (PC_{0-7})←addr	
	JNC addr	E6 address	2/2	jump to addr if C = 0	if C = 0 : (PC_{0-7})←addr	
	JZ addr	C6 address	2/2	jump to addr if A = 0	if A = 0 : (PC_{0-7})←addr	
	JNZ addr	96 address	2/2	jump to addr if A is NOT zero	if A≠0 : (PC_{0-7})←addr	
	JT0 addr	36 address	2/2	jump to addr if T0 = 0	if T0 = 0: (PC_{0-7})←addr	
	JNT0 addr	26 address	2/2	jump to addr if T0 = 1	if T0 = 1: (PC_{0-7})←addr	
	JT1 addr	56 address	2/2	jump to addr if T1 = 1	if T1 = 1: (PC_{0-7})←addr	
	JNT1 addr	46 address	2/2	jump to addr if T1 = 0	if T1 = 0: (PC_{0-7})←addr	
	JTF addr	16 address	2/2	jump to addr if Timer Flag = 1	if TF = 1: (PC_{0-7})←addr	4
	JNTF addr	06 address	2/2	jump to addr if Timer Flag = 0	if TF = 0: (PC_{0-7})←addr	

TIMER/EVENT COUNTER	MOV A, T	42	1/1	move timer/event counter contents to accumulator	(A)←(T)	
	MOV T, A	62	1/1	move accumulator contents to timer/event counter	(T)←(A)	
	STRT CNT	45	1/1	start event counter		
	STRT T	55	1/1	start timer		
	STOP TCNT	65	1/1	stop timer/event counter		
	EN TCNTI	25	1/1	enable timer/event counter interrupt		
	DIS TCNTI	35	1/1	disable timer/event counter interrupt		
CONTROL	EN I	05	1/1	enable external interrupt		
	DIS I	15	1/1	disable external interrupt		
	SEL RB0	C5	1/1	select register bank 0	(RBS)←0	5
	SEL RB1	D5	1/1	select register bank 1	(RBS)←1	5
	SEL MB0	E5	1/1	select program memory bank 0	(MBFF0)←0, (MBFF1)←0	
	SEL MB1	F5	1/1	select program memory bank 1	(MBFF0)←1, (MBFF1)←0	
	SEL MB2	A5	1/1	select program memory bank 2	(MBFF0)←0, (MBFF1)←1	
	SEL MB3	B5	1/1	select program memory bank 3	(MBFF0)←1, (MBFF1)←1	
	STOP	22	1/1	enter STOP mode		
	IDLE	01	1/1	enter IDLE mode		
SUBROUTINE	CALL addr	▲ 4 address	2/2	jump to subroutine	((SP))←(PC), ($PSW_{4,6,7}$) 6 (SP)←(SP) + 1 (PC_{8-10})←$addr_{8-10}$ (PC_{0-7})←$addr_{0-7}$ (PC_{11-12})←MBFF 0–1	6
	RET	83	1/2	return from subroutine	(SP)←(SP) − 1 (PC)←((SP))	6
	RETR	93	1/2	return from interrupt and restore bits 4, 6, 7 of PSW	(SP)←(SP) − 1 ($PSW_{4,6,7}$) + (PC)←((SP))	6

	mnemonic	opcode (hex.)	bytes/ cycles	description	function	notes
PARALLEL INPUT/OUTPUT	IN A, Pp	08 09 0A	1/2	input port p data to accumulator	(A)←(P0) (A)←(P1) (A)←(P2)	7
	OUTL Pp, A	38 39 3A	1/2	output accumulator data to port p	(P0)←(A) (P1)←(A) (P2)←(A)	
	ANL Pp, #data	98 99 9A	2/2	AND port p data with immediate data	(P0)←(P0) AND data (P1)←(P1) AND data (P2)←(P2) AND data	
	ORL Pp, #data	88 89 8A	2/2	OR port p data with immediate data	(P0)←(P0) OR data (P1)←(P1) OR data (P2)←(P2) OR data	
SERIAL INPUT/OUTPUT	MOV A, S_n	0C 0D	1/2	move serial I/O register contents to accumulator	(A)←(S0) (A)←(S1)	8
	MOV S_n, A	3C 3D 3E	1/2	move accumulator contents to serial I/O register	(S0)←(A) (S1)←(A) (S2)←(A)	9
	MOV S_n, #data	9C 9D 9E	2/2	move immediate data to serial I/O register	(S0)←data (S1)←data (S2)←data	
	EN SI	85	1/1	enable serial I/O interrupt		
	DIS SI	95	1/1	disable serial I/O interrupt		
	NOP	00	1/1	no operation		

Notes to Table 8

1. PSW CY, AC affected
2. PSW CY affected
3. PSW PS affected
4. Execution of JTF and JNTF instructions resets the Timer Flag (TF).

* : 8, 9, A, B, C, D, E, F
● : 0, 2, 4, 6, 8, A, C, E
▲ : 1, 3, 5, 7, 9, B, D, F

5. PSW RBS affected
6. PSW SP_0, SP_1, SP_2 affected
7. (A) = 1111 P23, P22, P21, P20.
8. (S1) has a different meaning for read and write operation, see serial I/O interface.
9. (S2) is a write only register. Reading S2 will give value FFH.

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Supply voltage (pin 28)	V_{DD}		−0,8 to + 8	V
All input voltages	V_I	0,8 to V_{DD} + 0,8		V
D.C. current into any input or output	$\pm I_I, \pm I_O$	max.	10	mA
Total power dissipation (see note)	P_{tot}	max.	500	mW
Power dissipation per output				
except P23, SCLK	P_O	max.	50	mW
P23, SCLK	P_O	max.	180	mW
Storage temperature range	T_{stg}		−65 to + 150	°C
Operating ambient temperature range	T_{amb}		−25 to + 70	°C
Operating junction temperature	T_j	max.	125	°C

Note

Thermal resistance (junction to ambient)				
for SOT-117D	$R_{th\ j\text{-}a}$	max.	120	K/W
for SOT-135A	$R_{th\ j\text{-}a}$	max.	60	K/W
for SOT-136A	$R_{th\ j\text{-}a}$	max.	150	K/W

DEVELOPMENT DATA

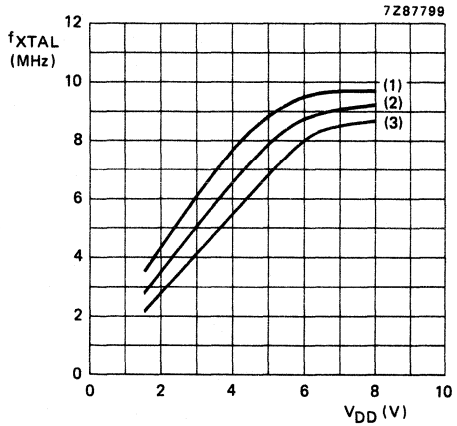
D.C. CHARACTERISTICS

V_{DD} = 2,75 to 6 V; V_{SS} = 0 V; T_{amb} = −25 to + 70 °C; all voltages with respect to V_{SS}; f = 3,58 MHz with R_S = 50 Ω; unless otherwise specified.

parameter	symbol	min.	typ.	max.	unit
Supply voltage					
operating (see Fig. 23)	V_{DD}	1,8	–	6	V
STOP mode for RAM retention	V_{DD}	1,0	–	6	V
Supply current					
operating					
at V_{DD} = 3 V (see Fig. 24)	I_{DD}	–	600	–	μA
IDLE mode					
at V_{DD} = 3 V (see Fig. 25)	I_{DD}	–	300	–	μA
STOP mode (see Fig. 26 and note 1)					
at V_{DD} = 1,8 V; T_{amb} = 25 °C	I_{DD}	–	1,2	2,5	μA
at V_{DD} = 1,8 V; T_{amb} = 55 °C	I_{DD}	–	–	5	μA
at V_{DD} = 1,8 V; T_{amb} = 70 °C	I_{DD}	–	–	10	μA
RESET I/O					
Switching level	V_{RESET}	–	1,3	–	V
Sink current					
at $V_{DD} > V_{RESET}$	I_{OL}	–	7	–	μA
Inputs					
Input voltage LOW	V_{IL}	0	–	$0,3V_{DD}$	V
Input voltage HIGH	V_{IH}	$0,7V_{DD}$	–	V_{DD}	V
Input leakage current					
at $V_{SS} < V_I < V_{DD}$	$\pm I_{IL}$	–	–	1	μA
Outputs					
Output voltage LOW					
at $V_I = V_{SS}$ or V_{DD}; $\|I_O\| < 1$ μA	V_{OL}	–	–	0,05	V
Output sink current LOW					
at V_{DD} = 3 V; V_O = 0,4 V					
except P23/SDA, SCLK (see Fig. 27)	I_{OL}	0,75	1,5	–	mA
P23/SDA, SCLK (see Fig. 28)	I_{OL}	1,5	–	–	mA
Pull-up output source current HIGH (see Fig. 29)					
at V_{DD} = 3 V; $V_O = 0,9V_{DD}$	$-I_{OH}$	25	–	–	μA
at V_{DD} = 3 V; $V_O = V_{SS}$	$-I_{OH}$	–	–	200	μA
Push-pull output source current HIGH					
at V_{DD} = 3 V; $V_O = V_{DD}-0,4$ V	$-I_{OH}$	0,75	1,5	–	mA

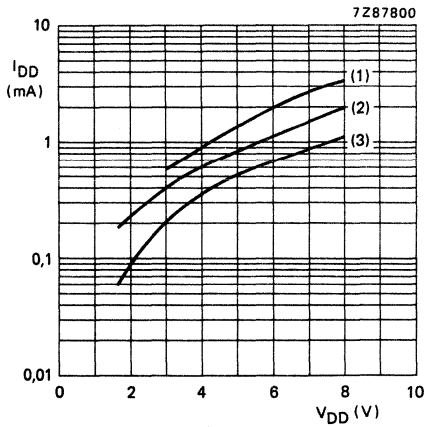
Note 1

Crystal connected between XTAL 1 and XTAL 2; SCL and SDA pulled to V_{DD} via 5,6 kΩ resistor; CE and T1 at V_{SS}.



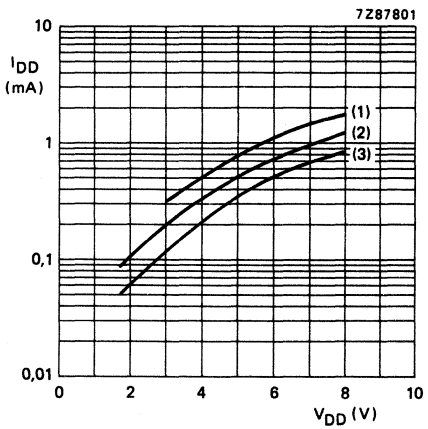
(1) $T_{amb} = -25\ ^{o}C$
(2) $T_{amb} = 25\ ^{o}C$
(3) $T_{amb} = 70\ ^{o}C$

Fig. 23 Maximum clock frequency (f_{XTAL}) as a function of the supply voltage (V_{DD}).



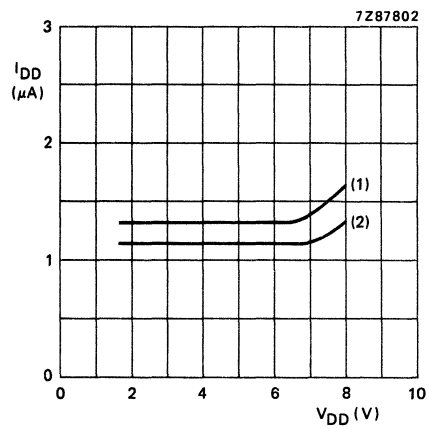
(1) clock frequency = 4 MHz
(2) clock frequency = 2 MHz
(3) clock frequency = 500 kHz

Fig. 24 Typical supply current (I_{DD}) in operating mode as a function of the supply voltage (V_{DD}); $T_{amb} = 25\ ^{o}C$.



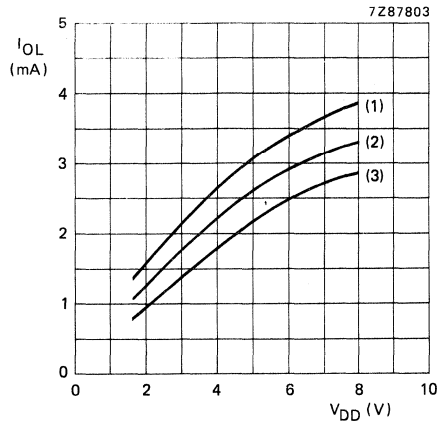
(1) clock frequency = 4 MHz
(2) clock frequency = 2 MHz
(3) clock frequency = 500 kHz

Fig. 25 Typical supply current (I_{DD}) in IDLE mode as a function of the supply voltage (V_{DD}); $T_{amb} = 25\ ^{o}C$.



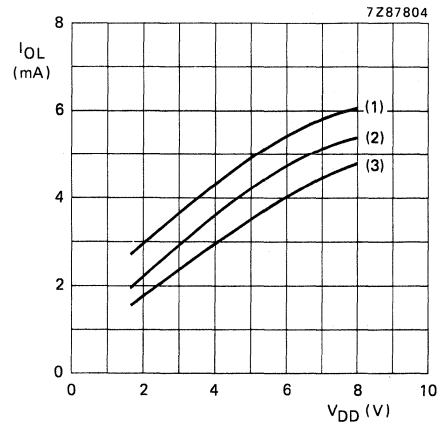
(1) $T_{amb} = 70\ ^{o}C$
(2) $T_{amb} = 25\ ^{o}C$

Fig. 26 Typical supply current (I_{DD}) in STOP mode as a function of the supply voltage (V_{DD}).



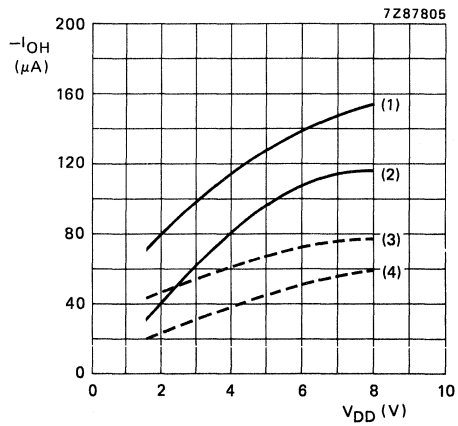
(1) $T_{amb} = -25\ ^{o}C$
(2) $T_{amb} = 25\ ^{o}C$
(3) $T_{amb} = 70\ ^{o}C$

Fig. 27 Output sink current LOW (I_{OL}), except outputs P23/SDA and SCLK, as a function of supply voltage (V_{DD}); V_O = 0,4 V.



(1) $T_{amb} = -25\ ^{o}C$
(2) $T_{amb} = +25\ ^{o}C$
(3) $T_{amb} = +70\ ^{o}C$

Fig. 28 Output current LOW (I_{OL}), outputs P23/SDA and SCLK, as a function of supply voltage (V_{DD}); V_O = 0,4 V.



(1) $T_{amb} = 25\ ^{o}C$; $V_O = V_{SS}$
(2) $T_{amb} = 25\ ^{o}C$; $V_O = 0{,}9V_{DD}$
(3) $T_{amb} = 70\ ^{o}C$; $V_O = V_{SS}$
(4) $T_{amb} = 70\ ^{o}C$; $V_O = 0{,}9V_{DD}$

Fig. 29 Output source current HIGH ($-I_{OH}$) as a function of supply voltage (V_{DD}).

A.C. CHARACTERISTICS

Rise and fall times between 10 and 90% levels; C_L = 50 pF

parameter	symbol	at 70 °C max. value			unit
	V_{DD}	1,8	3,0	6,0	V
Fall time	t_f	200	100	70	ns
Rise time	t_r	200	100	80	ns

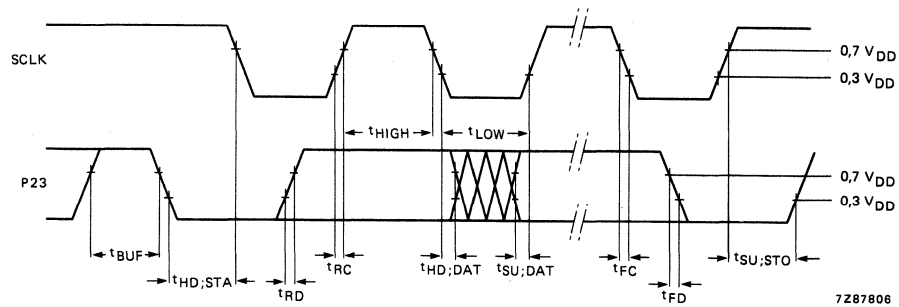


Fig. 30 PCD3343 timing requirements for the P23 and SCLK *input* signals.

Table 9 Input timing shown in figure 30

symbol	timing
t_{BUF}	$\geqslant 14t_{XTAL}$
$t_{HD;STA}$	$\geqslant 14t_{XTAL}$
t_{HIGH}	$\geqslant 17t_{XTAL}$
t_{LOW}	$\geqslant 17t_{XTAL}$
$t_{SY;STO}$	$\geqslant 14t_{XTAL}$
$t_{HD;DAT}$	> 0
$t_{SU;DAT}$	$\geqslant 250$ ns
t_{RD}	$\leqslant 1\ \mu s$
t_{RC}	$\leqslant 1\ \mu s$
t_{FD}	$\leqslant 1\ \mu s$
t_{FC}	$\leqslant 0{,}3\ \mu s$

Notes to Table 9

t_{XTAL} = one period of the XTAL input frequency (f_{XTAL})
= 280 ns for f_{XTAL} = 3,58 MHz.
These figures apply to all modes.

DEVELOPMENT DATA

A.C. CHARACTERISTICS (continued)

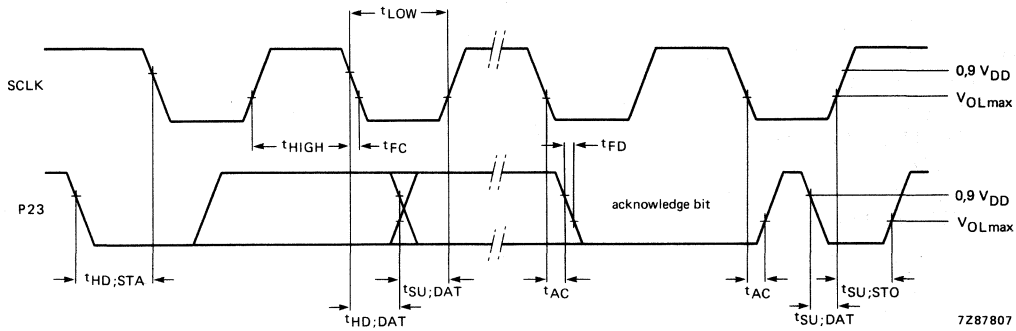


Fig. 31 PCD3343 timing requirements for the P23 and SCLK *output* signals.

Table 10 Output timing shown in figure 31

symbol	timing	
	normal mode (ASC in S2 = 0)	low-speed mode (ASC in S2 = 1)
$t_{HD;STA}$	½ (DF + 9) t_{XTAL}	¾ (DF + 9) t_{XTAL}
t_{HIGH}	½ (DF) t_{XTAL}	¾ (DF) t_{XTAL}
t_{LOW}	½ (DF) t_{XTAL}	¼ (DF) t_{XTAL}
$t_{SU;STO}$	½ (DF − 3) t_{XTAL}	¼ (DF − 3) t_{XTAL}
$t_{HD;DAT}$ (slave transmitter		
any DF	$\geqslant 9t_{XTAL}$	$\geqslant 9t_{XTAL}$
	$\leqslant 12t_{XTAL}$	$\leqslant 12t_{XTAL}$
$t_{HD;DAT}$ (master transmitter)		
for DF ⩽ 51	$\geqslant 9t_{XTAL}$	–
	$\leqslant 12t_{XTAL}$	–
for DF ⩽ 99	–	$\geqslant 9t_{XTAL}$
	–	$\leqslant 12t_{XTAL}$
$t_{SU;DAT}$ (master transmitter)		
for DF > 51	$\geqslant 15t_{XTAL}$	–
	$\leqslant 24t_{XTAL}$	–
for DF > 99	–	$\geqslant 15t_{XTAL}$
for DF ⩽ 51	$\geqslant 9t_{XTAL}$	$\leqslant 24t_{XTAL}$
for DF ⩽ 99	–	$\geqslant 9t_{XTAL}$
t_{AC}	$\geqslant 9t_{XTAL}$	$\geqslant 9t_{XTAL}$
	$\leqslant 12t_{XTAL}$	$\leqslant 12t_{XTAL}$
t_{FD}, t_{FC}	⩽ 100 ns at C_b = 400 pF	⩽ 100 ns at C_b = 400 pF

Notes to Table 10

t_{XTAL} = one period of the XTAL input frequency (f_{XTAL})
= 280 ns for f_{XTAL} = 3,58 MHz.
DF = divisor (see Table 2 Serial I/O section).
C_b = the maximum bus capacitance for each line.

DEVELOPMENT DATA

APPLICATION INFORMATION

A block diagram of an electronic featurephone built around the PCD3343 is shown in figure 32. It comprises the following dedicated telephony IC's:

- TEA1060/1061 transmission circuit for telephony
- PCD3312 DTMF generator with Serial I/O
- PCE2111 or PCF8577 2 LCD drivers in LCD module MB7020160
- PCD8571 1 K RAM's with Serial I/O; the number of RAM's depends on the required amount of stored telephone numbers
- PCD3360/3361 programmable multi-tone ringer

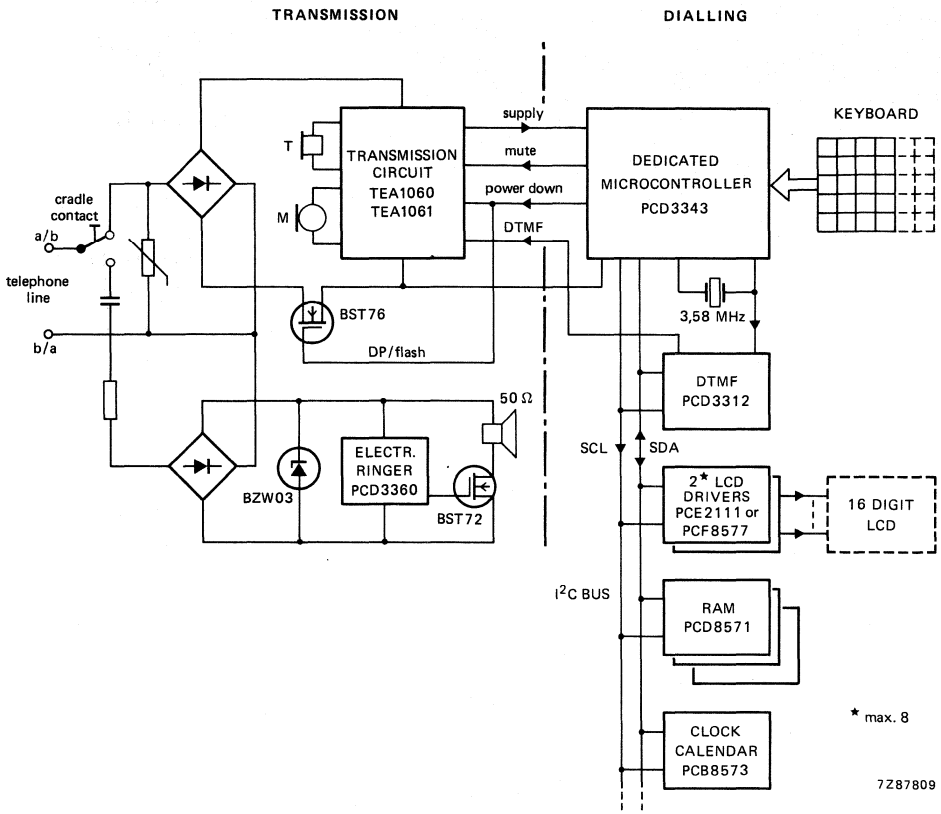


Fig. 32 Block diagram of electronic featurephone with common line interface.

A detailed application diagram of the PCD3343 with PCD3312 (DTMF), two PCD8571 (RAM) and two PCE2111 (LCD display drivers) is shown in figure 33.

Row 5 of the keyboard contains the following special keys:

- P program and autodial
- FL flash or register recall
- R redial or extended redial
- AP access pause

Row 6 contains the different diode options.

Columns 5 and 6 contain the button keys M0 to M9; single name keys for repertory telephone numbers.

APPLICATION INFORMATION (continued)

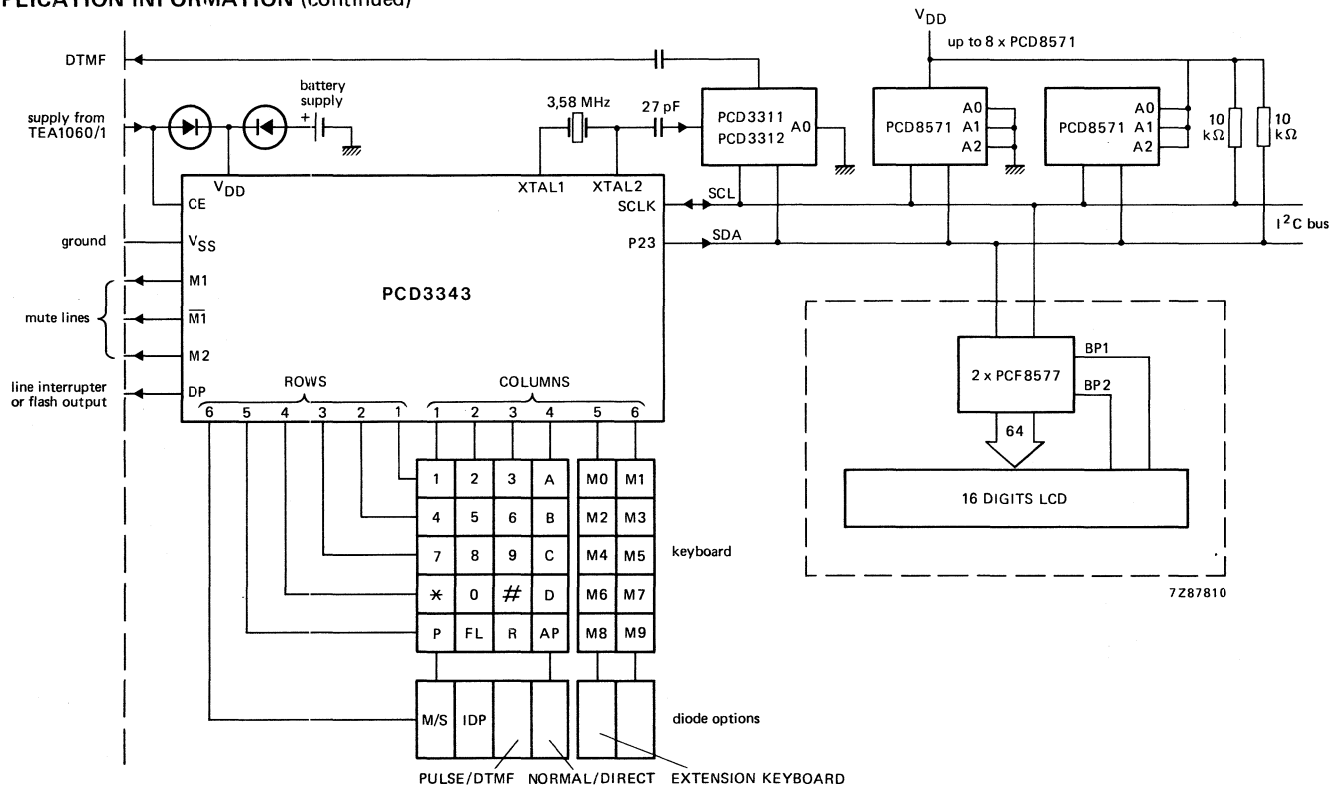


Fig. 33 Application diagram of PCD3343 for electronic featurephone with associated keyboard.

Additional information is available on request for the following:

- Serial I/O
- I^2C bus specification
- Interrupt logic
- Instruction set descriptions
- Software routines for an intelligent telephone set

16-BIT MICROPROCESSOR FAMILY SC68000

Microprocessor unit

16-BIT MICROPROCESSOR

Originally published by Signetics May 1983

Preliminary

DESCRIPTION

The Signetics SCN68000 combines state-of-the-art semiconductor technology and advanced circuit design techniques with computer sciences to achieve an architecturally advanced 16-bit microprocessing unit. As shown in the programming model, figure 1, the SCN68000 offers seventeen 32-bit registers in addition to a 32-bit program counter and a 16-bit status register The first eight registers (D0–D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The second set of seven registers (A0–A6) and the system stack pointer can be used as software stack pointers and base address registers. In addition, these registers can be used for word and long word address operations. All 17 registers can be used as index registers.

FEATURES

- **32-bit data and address registers**
- **16 megabyte direct addressing range**
- **56 powerful instruction types**
- **Operations on five main data types**
- **Memory mapped I/O**
- **14 addressing modes**
- **Multilevel interrupt structure**
- **Signed and unsigned multiply and divide**
- **User and supervisor modes**
- **Support for high level languages**
- **Testing and debugging support**

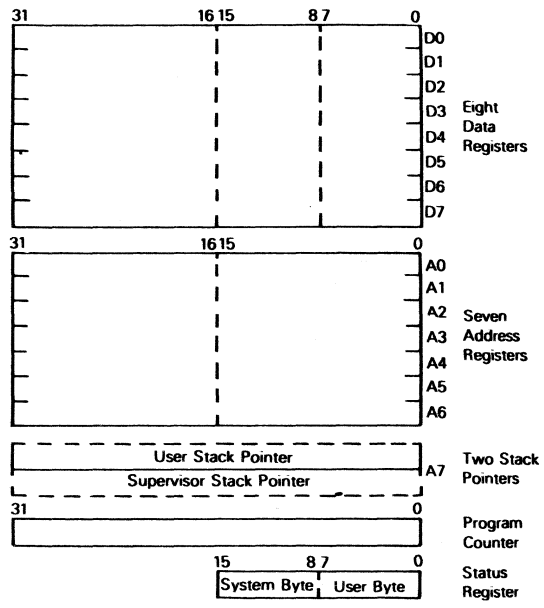
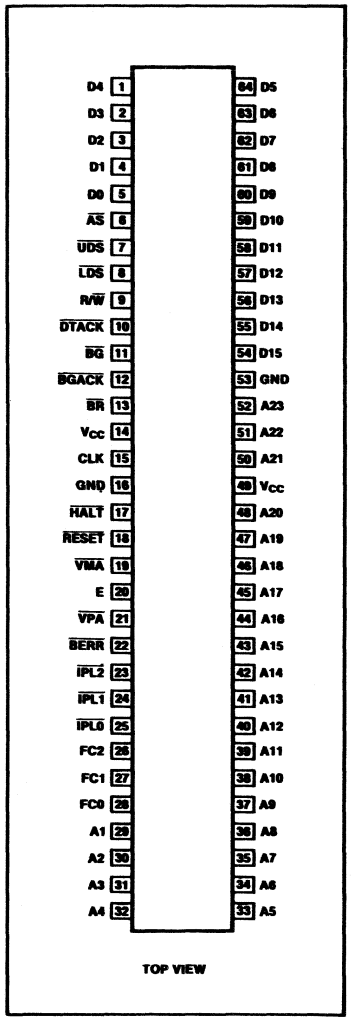


Figure 1. Programming Model

PIN CONFIGURATION[1]



TOP VIEW

ORDERING CODE

PACKAGES	V_{DD} = 5V ±5%, T_A = 0° to 70°C			
	4MHz	6MHz	8MHz	10MHz
Ceramic DIP	SCN68000C4I64	SCN68000C6I64	SCN68000C8I64	SCN68000CAI64
Plastic DIP	SCN68000C4N64	SCN68000C6N64	SCN68000C8N64	SCN68000CAN64

[1] In this data sheet, barring signal names (overscore) to indicate low is done only for the pin configuration diagram, signal description headings, tables and figures.

Preliminary

SIGNAL DESCRIPTION

The input and output signals can be functionally organized into the groups shown in figure 2. The I/O signal characteristics are summarized in table 1.

Address Bus (A1-A23)

This 23-bit, unidirectional, three-state bus is capable of addressing eight megawords of data. It provides the address for bus operation during all cycles except interrupt cycles. During interrupt cycles, address lines A1, A2, and A3 provide information about what level interrupt is being serviced while address lines A4-A23 are all set to a logic high.

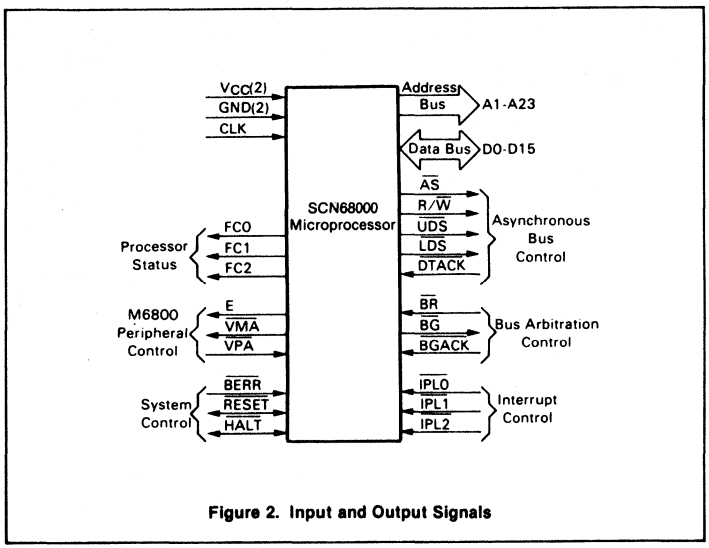


Figure 2. Input and Output Signals

Data Bus (D0-D15)

This 16-bit, bidirectional, three state bus is the general purpose data path. It can transfer and accept data in either word or byte length. During an interrupt acknowledge cycle, an external device supplies the interrupt vector on data lines D0-D7.

Asynchronous Bus Control

Asynchronous data transfers are handled using the following control signals:

Address Strobe ($\overline{AS}$) — This signal indicates that there is a valid address on the address bus.

Read/Write (R/$\overline{W}$) — This signal defines the data bus transfer as a read or write cycle. The R/W signal also works in conjunction with the upper and lower data strobes as explained in the next paragraph.

Upper and Lower Data Strobes ($\overline{UDS}$, $\overline{LDS}$) — These signals control the data on the bus as shown in table 2. When the R/W line is high, the processor will read from the data bus as indicated. When the R/W line is low, the processor will write to the data bus as shown.

Data Transfer Acknowledge ($\overline{DTACK}$) — This input indicates that the data transfer is completed. When the processor recognizes DTACK during a read cycle, data is latched and the bus cycle is terminated. When DTACK is recognized during a write cycle, the bus cycle is terminated. An active transition of DTACK indicates the termination of a data transfer on the bus.

If the system must run at a maximum rate determined by RAM access times, the relationship between the times at which DTACK and data are sampled is important. All control and data lines are sampled during the SCN68000's clock high time. The clock is internally buffered, which results in some slight differences in the sampling and recognition of various signals. The DTACK signal, like other control signals, is internally synchronized to allow for valid operation in an asynchronous system. If the required setup time (#47)[1] is met during S4, DTACK will be recognized during

Table 1 SIGNAL SUMMARY

Signal Name	Mnemonic	Input/Output	Active State	Three State
Address Bus	A1-A23	output	high	yes
Data Bus	D0-D15	input/output	high	yes
Address Strobe	$\overline{AS}$	output	low	yes
Read/Write	R/$\overline{W}$	output	read-high write-low	yes
Upper and Lower Data Strobes	$\overline{UDS}$, $\overline{LDS}$	output	low	yes
Data Transfer Acknowledge	$\overline{DTACK}$	input	low	—
Bus Request	$\overline{BR}$	input	low	—
Bus Grant	$\overline{BG}$	output	low	no
Bus Grant Acknowledge	$\overline{BGACK}$	input	low	—
Interrupt Priority Level	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	input	low	—
Bus Error	$\overline{BERR}$	input	low	—
Reset	$\overline{RESET}$	input/output	low	no*
Halt	$\overline{HALT}$	input/output	low	no*
Enable	E	output	high	—
Valid Memory Address	$\overline{VMA}$	output	low	yes
Valid Peripheral Address	$\overline{VPA}$	input	low	—
Function Code Output	FC0, FC1, FC2	output	high	yes
Clock	CLK	input	high	no
Power Input	V_{CC}	input	—	—
Ground	GND	input	—	—

*open drain

[1] Number references (#) can be found in the AC Electrical Characteristics section and corresponding timing diagrams located towards the end of this data sheet.

Preliminary

Table 2 DATA STROBE CONTROL OF DATA BUS

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8-D15	D0-D7
High	High	–	No valid data	No valid data
Low	Low	High	Valid data bits 8-15	Valid data bits 0-7
High	Low	High	No valid data	Valid data bits 0-7
Low	High	High	Valid data bits 8-15	No valid data
Low	Low	Low	Valid data bits 8-15	Valid data bits 0-7
High	Low	Low	Valid data bits 0-7*	Valid data bits 0-7
Low	High	Low	Valid data bits 8-15	Valid data bits 8-15*

*These conditions are a result of current implementation and may not appear on future devices.

S5 and S6, and data will be captured during S6. The data must meet the required setup time (#27). If an asynchronous control signal does not meet the required setup time, it is possible that it may not be recognized during that cycle. Because of this, asynchronous systems must not allow DTACK to precede data by more than parameter #31.

Asserting DTACK (or BERR) on the rising edge of a clock (such as S4) after the assertion of address strobe will allow an SCN68000 system to run at its maximum bus rate. If setup times #27 and #47 are guaranteed, #31 may be ignored.

Bus Arbitration Control

These three signals form a bus arbitration circuit to determine which device will be the bus master device:

Bus Request ($\overline{BR}$) — This input is wire ORed with all other devices that could be bus masters. It indicates to the processor that some other device desires to become the bus master.

Bus Grant ($\overline{BG}$) — This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

Bus Grant Acknowledge ($\overline{BGACK}$) — This input indicates that some other device has become the bus master. This signal cannot be asserted until the following four conditions are met:

1. A bus grant has been received.
2. Address strobe is inactive, indicating that the microprocessor is not using the bus.
3. Data transfer acknowledge is inactive, indicating that another device is not using the bus.
4. Bus grant acknowledge is inactive, indicating that no other device is still claiming bus mastership.

Interrupt Control ($\overline{IPL0}$,$\overline{IPL1}$,$\overline{IPL2}$)

These inputs indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. The least significant bit is given in IPL0 and the most signficant bit is contained in IPL2.

System Control

The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred.

Bus Error ($\overline{BERR}$) — This input informs the processor that there is a problem with the cycle currently being executed. Problems may be the result of nonresponding devices, interrupt vector acquisition failure, illegal access request as determined by a memory management unit, or other application dependent errors. The bus error signal interacts with the halt signal to determine if exception processing should be performed or the current bus cycle should be retried (see Bus Error and Halt Operation for additional information).

Reset ($\overline{RESET}$) — This bidirectional signal line acts to reset the processor (initiate a system initialization sequence) in response to an external reset signal. An internally generated reset (result of a RESET instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external HALT and RESET signals applied at the same time (see Reset Operation for additional information).

Halt ($\overline{HALT}$) — When this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state. When the processor has stopped executing instructions, such as in a double bus fault condition, the halt line is driven by the processor to indicate to external devices that the processor has stopped (see Bus Error and Halt Operation for additional information).

Peripheral Control

These control signals are used to allow the interfacing of synchronous peripheral devices with the asynchronous SCN68000:

Enable (E) — This signal is the enable signal for synchronous type peripheral devices. The period for this output is ten SCN68000 clock periods (six clocks low; four clocks high).

Valid Peripheral Address ($\overline{VPA}$) — This input indicates that the device or region addressed is a synchronous device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt (see Interface with Synchronous Peripherals for additional information).

Valid Memory Address ($\overline{VMA}$) — This output is used to indicate to synchronous peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable. This signal is issued only in response to a valid peripheral address (VPA) input which indicates that the peripheral is a synchronous device.

Processor Status (FC0,FC1,FC2)

These function code outputs indicate the state (user or supervisor) and the cycle type currently being executed (see table 3). The information indicated by the function code is valid whenever address strobe (AS) is active.

Preliminary

Table 3 FUNCTION CODE OUTPUTS

FC2	FC1	FC0	Cycle Type
Low	Low	Low	(Undefined, Reserved)
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	(Undefined, Reserved)
High	Low	Low	(Undefined, Reserved)
High	Low	High	Supervisor Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

Clock (CLK)

The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input is a constant frequency.

REGISTER DESCRIPTION AND DATA ORGANIZATION

Operand Size

Operand sizes are defined as follows: a byte equals 8-bits,a word equals 16-bits, and a long word equals 32-bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. All explicit instructions support byte, word or long word operands. Implicit instructions support some subset of all three sizes.

Data Organization in Registers

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers together with the active stack pointer support address operands of 32 bits.

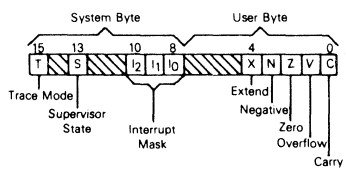
Data Registers — Each data register is 32-bits wide. Byte operands occupy the low order 8-bits, word operands the low order 16-bits, and long word operands the entire 32-bits. The least significant bit is addressed as bit 0; the most significant bit is addressed as bit 31. When a data register is used as either a source or destination operand and the operand size is not 32 bits, only the appropriate low-order portion is changed; the remaining high-order portion is neither used nor changed.

Address Registers — Each address register and the stack pointer are 32-bits wide and hold a full 32-bit address. Address registers do not support byte sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending on the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32-bits before the operation is performed.

Status Register

The status register contains the interrupt mask (eight levels available) as well as the condition codes: extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in a trace (T) mode and/or in a supervisor (S) state.

STATUS REGISTER



Data Organization in Memory

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in figure 3. The low order byte has an odd address that is one count higher than the word address. Instructions and multibyte data are accessed only on word (even byte) boundaries. If a long word data is located at address n (n even), then the second word of that data is located at address n + 2.

The data types supported by the SCN68000 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory as shown in figure 4.

BUS OPERATION

The following is an explanation of control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

Data Transfer Operations

Transfer of data between devices involves address bus A1-A23, data bus D0-D15 and control signals. The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

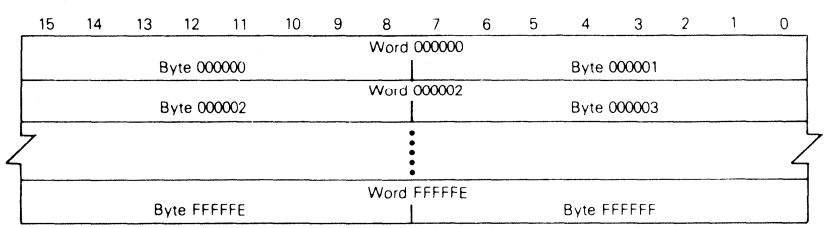


Figure 3. Word Organization in Memory

Preliminary

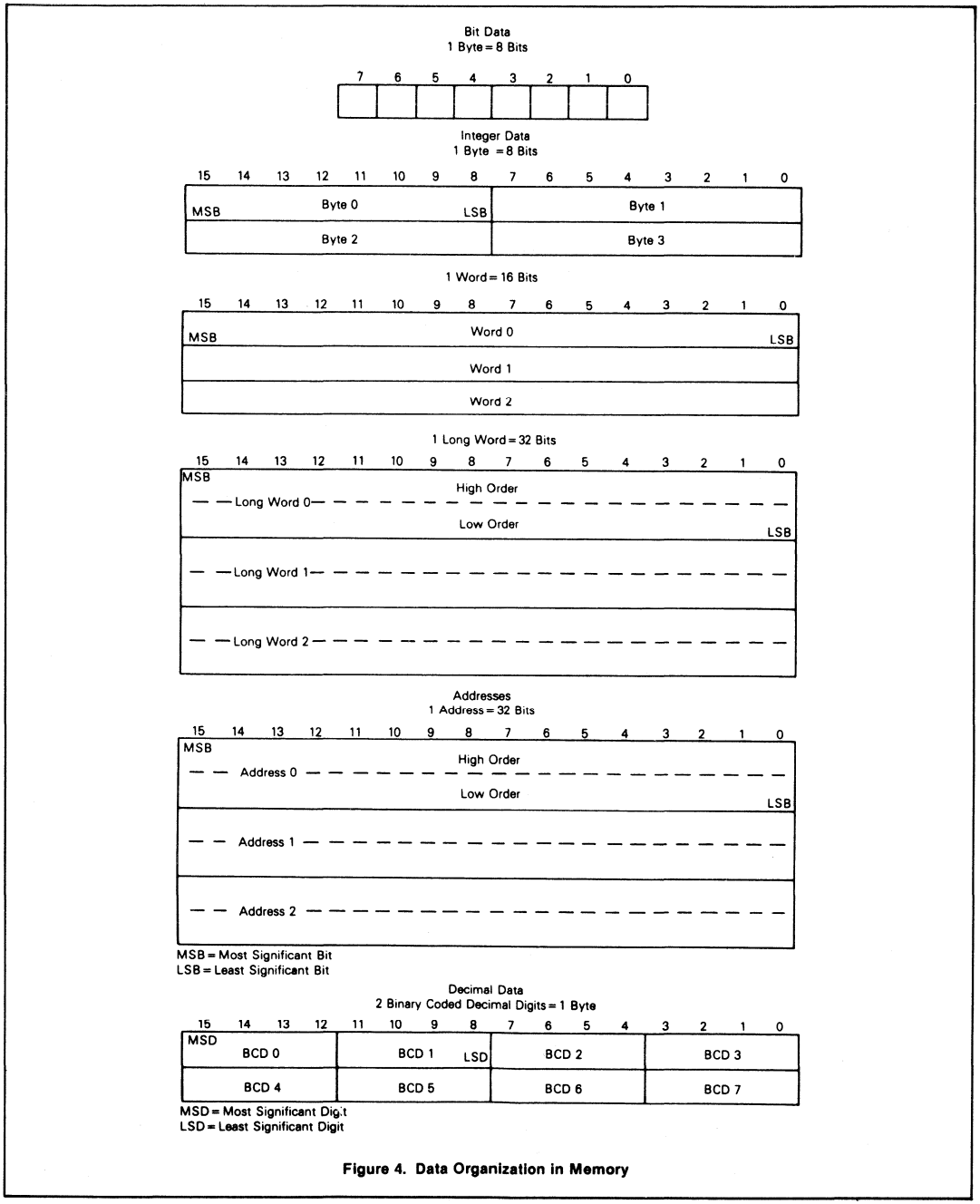


Figure 4. Data Organization in Memory

Preliminary

NOTE
The terms assertion and negation are used extensively to avoid confusion when dealing with a mixture of 'active low' and 'active high' signals. Assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is low or high. Negate or negation is used to indicate that a signal is inactive or false.

Read Cycle — During a read cycle, the processor receives data from memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or double word) operation, the processor reads both bytes simultaneously. When the instruction specifies byte operation, the processor uses an internal A0 bit to determine which byte to read and then issues the data strobe required for that byte: when the A0 bit equals zero, the upper data strobe is issued; when the A0 bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally.

Flow charts for word and byte read cycles are shown in figures 5 and 6 respectively. Figure 7 illustrates read and write cycle timing and figure 8 illustrates the timing for word and byte read cycles.

Write Cycle — During a write cycle, the processor sends data to memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes simultaneously. When the instruction specifies a byte operation, the processor uses an internal A0 bit to determine which byte to write and then issues the data strobe required for that byte: when the A0 bit equals zero, the upper data strobe is issued; when the A0 bit equals one, the lower data strobe is issued.

Flow charts for word and byte write cycles are shown in figures 9 and 10 respectively. Figure 11 illustrates the timing for both cases.

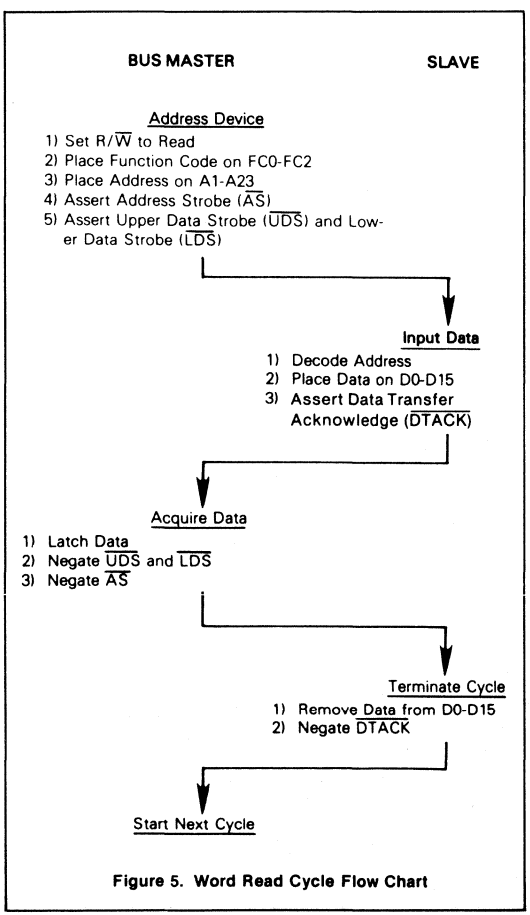


Figure 5. Word Read Cycle Flow Chart

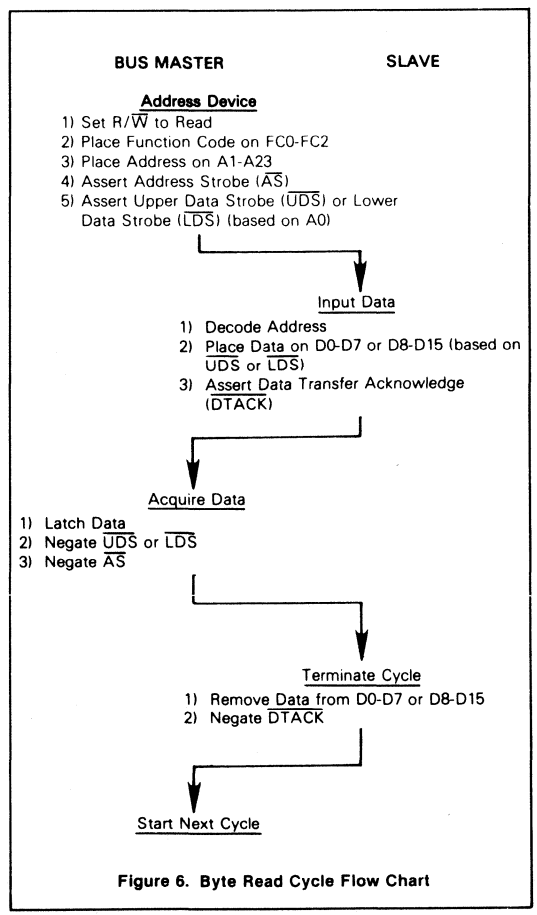


Figure 6. Byte Read Cycle Flow Chart

Preliminary

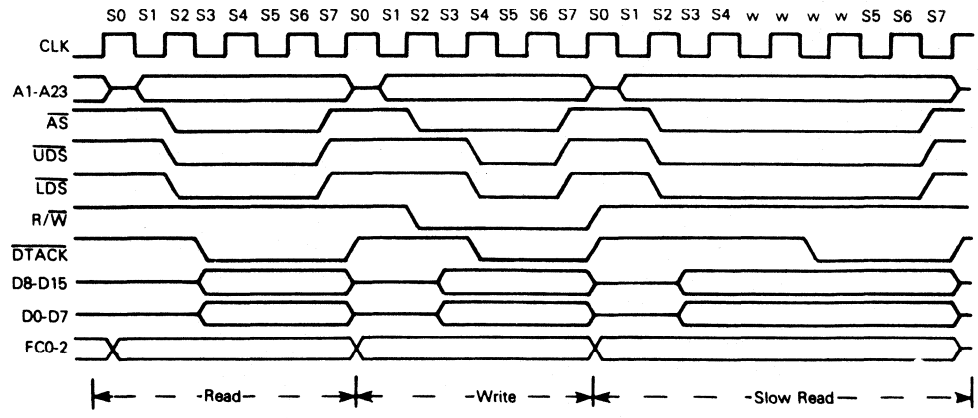


Figure 7. Read and Write Cycle Timing

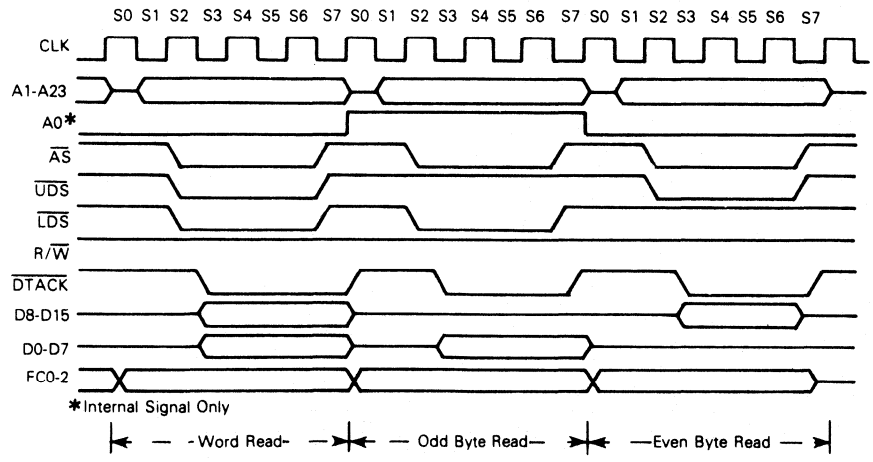


Figure 8. Word and Byte Read Cycle Timing

Preliminary

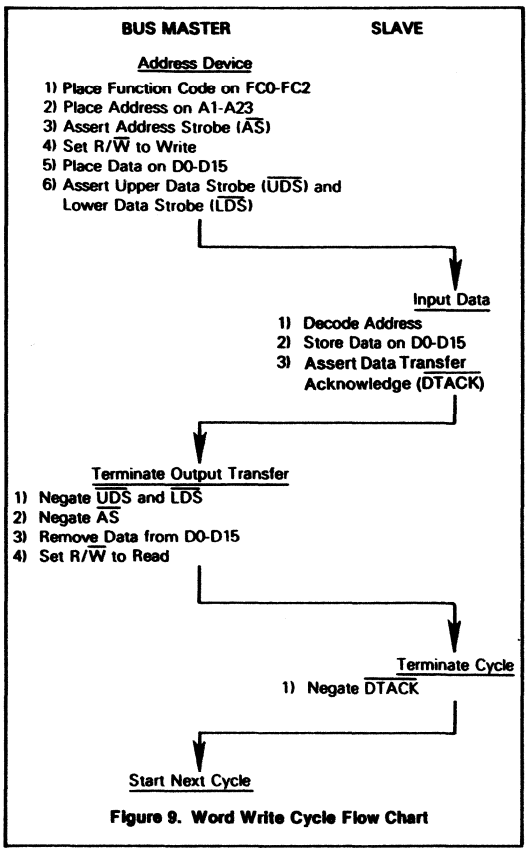


Figure 9. Word Write Cycle Flow Chart

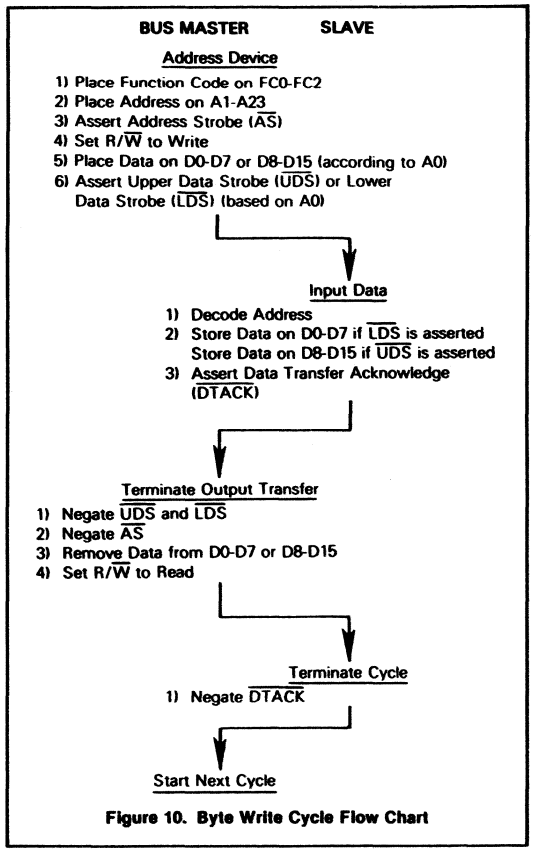


Figure 10. Byte Write Cycle Flow Chart

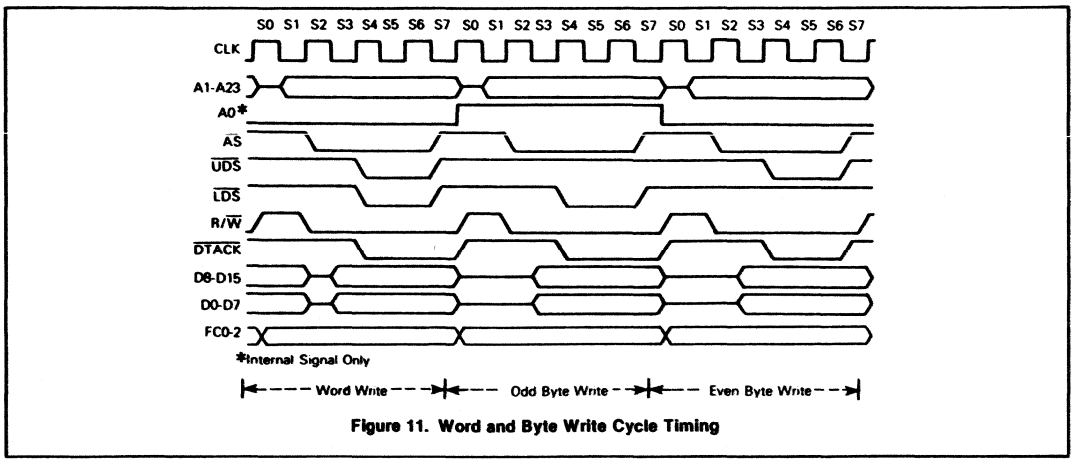


Figure 11. Word and Byte Write Cycle Timing

Preliminary

Read-Modify-Write Cycle — The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the SCN68000, this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment, and is the only instruction that uses it. The read-modify-write cycle is always a byte operation. The flow chart is given in figure 12 and a timing diagram is shown in figure 13.

Bus Arbitration

Bus arbitration is a technique used by master type devices to request, be granted, and acknowledge bus mastership. In its simplist form, it consists of:

1. Asserting a bus mastership request.
2. Receiving a grant that the bus is available at the end of the current cycle.
3. Acknowledging that mastership has been assumed.

Figure 14 is a flow chart showing the detail involved in a request from a single device. Figure 15 is a timing diagram for the same operations. The technique used allows processing of bus requests during data transfer cycles.

The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of mastership, the bus request line from each device is wire-ORed to the processor. In this system, there could be more than

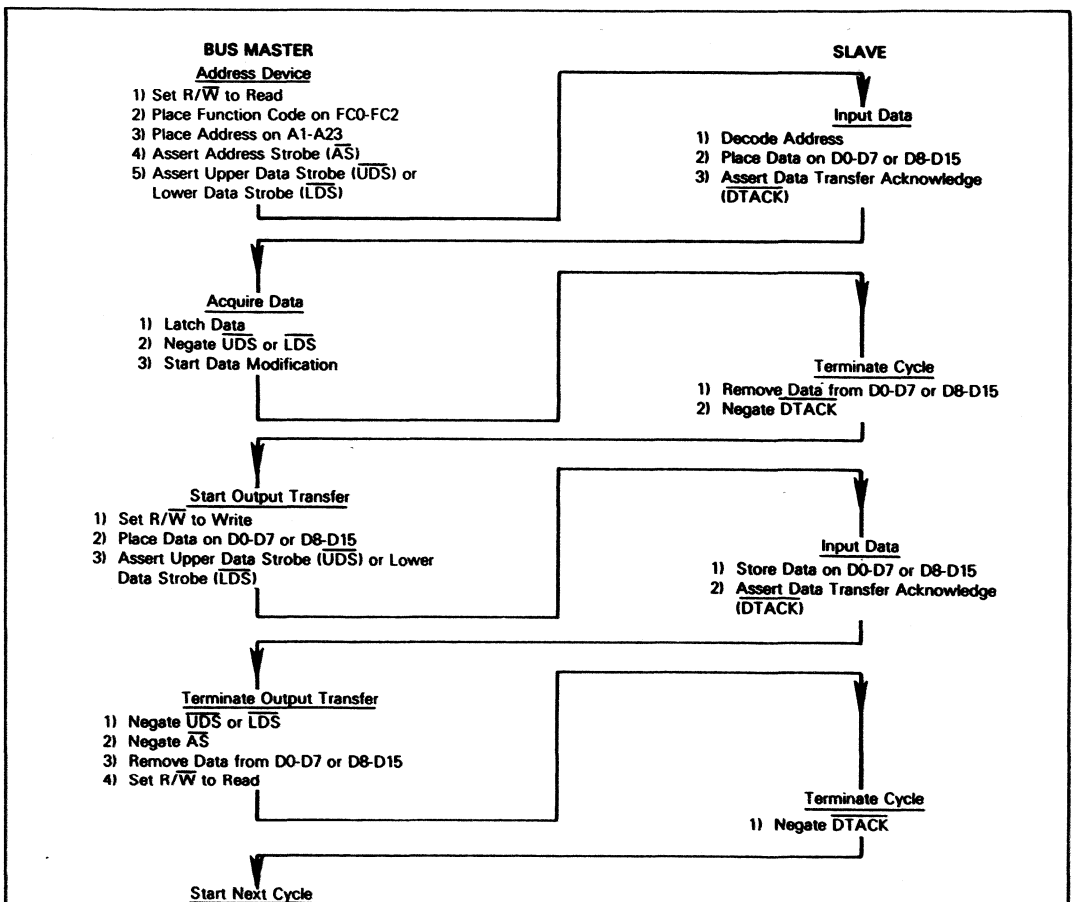


Figure 12. Read-Modify-Write Cycle Flow Chart

Preliminary

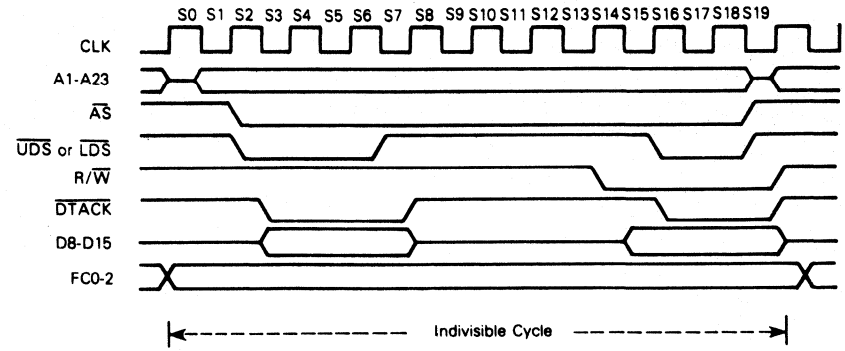


Figure 13. Read-Modify-Write Cycle Timing

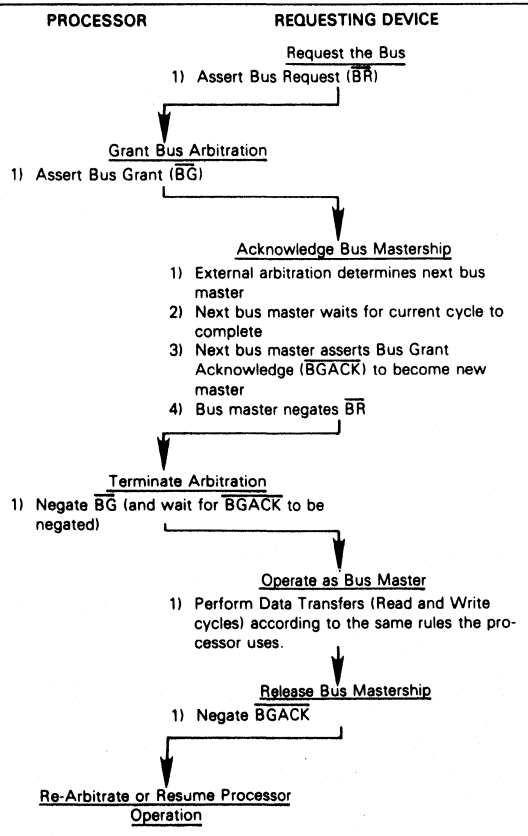


Figure 14. Bus Arbitration Cycle Flow Chart

one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge (BGACK) signal. However, if the bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements.

Requesting the Bus — External devices, capable of becoming bus masters, request the bus by asserting the bus request (BR) signal. This is a wire-ORed signal (although it need not be constructed from open collector devices) that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level than the external device and will relinquish the bus after it has completed the last bus cycle it has started.

When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry inadvertently responded to noise.

Receiving the Bus Grant — The processor asserts bus grant (BG) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe (AS) signal. In this case, bus grant will not be asserted until one

Preliminary

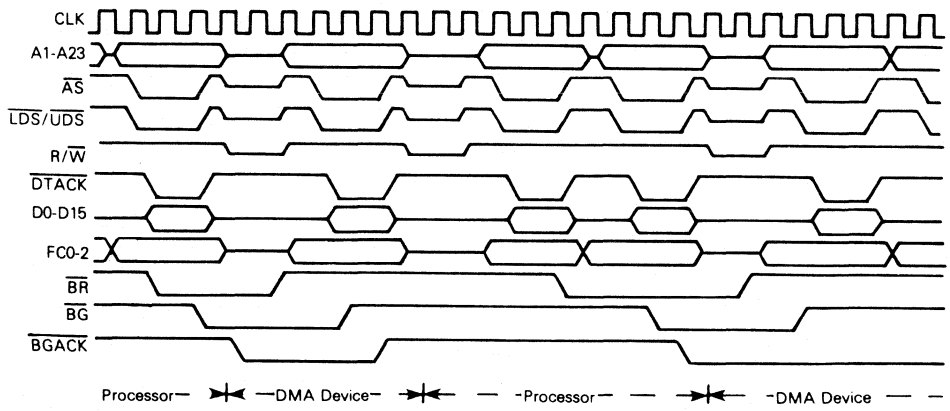


Figure 15. Bus Arbitration Cycle Timing

clock after address strobe is asserted to indicate to external devices that a bus cycle is being executed.

The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

Acknowledgement of Mastership — Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own BGACK. The negation of the address strobe indicates that the previous master has completed its cycle, and the negation of bus grant acknowledge indicates that the previous master has released the bus. A device is not allowed to 'break into' a cycle while address strobe is asserted. The negation of data transfer acknowledge indicates that the previous slave has terminated its connection to the previous master. Note that in some applications, data transfer acknowledge might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledge is issued, the device remains the bus master until it negates bus grant acknowledge. Bus grant acknowledge should not be negated until after the bus cycle(s) is completed.

The bus request from the granted device should be dropped when bus grant acknowledge is asserted. If bus request is still asserted after bus grant acknowledge is negated, the processor performs another arbitration sequence and issues another bus grant. Note that the processor does not perform any external bus cycles before it reasserts bus grant.

Bus Arbitration Control—The bus arbitration control unit in the SCN68000 is implemented with a finite state machine whose state diagram is shown in figure 16. All asynchronous signals to the SCN68000 are synchronized before being used internally. This synchronization is accomplished in a maximum of one cycle of the system clock, assuming that the asynchronous input setup time (#47) has been met (see figure 17). The input signal is sampled on the falling edge of the clock and is valid internally after the next falling edge.

As shown in figure 16, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins respectively. The bus grant output is labeled G and the internal three-state control signal T. If T is true, the address, data, and control buses are placed in a high-impedance state when AS is negated. All signals are shown in positive logic (active high) regardless of their true active voltage level.

State changes (valid outputs) occur on the next rising edge after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in figure 18. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in figure 19.

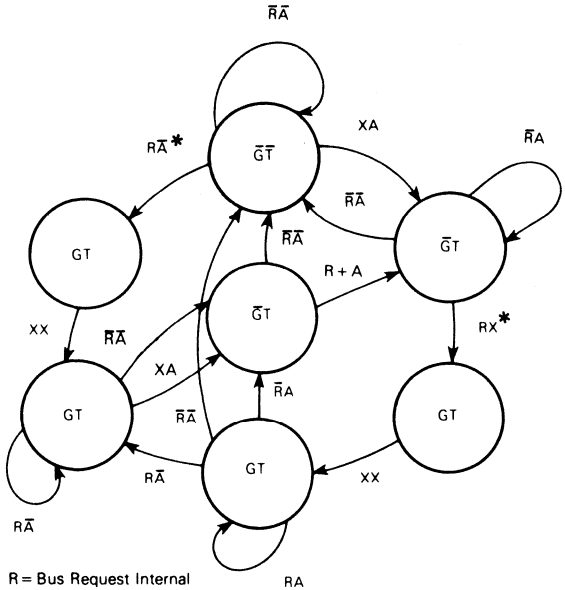
If a bus request is made at a time when the MPU has already begun a bus cycle but AS has not been asserted (bus state S0), BG will not be asserted on the next rising edge following its internal assertion. This sequence is shown in figure 20.

Bus Error and Halt Operation

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error signal is received, the processor has the option of either initiating a bus error exception sequence or trying to run the bus cycle again.

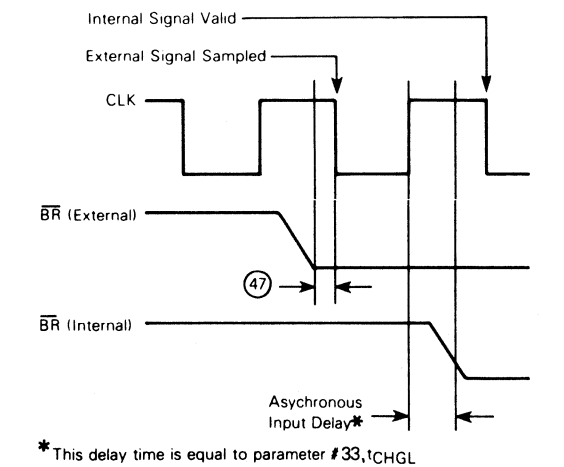
Exception Sequence — When the bus error signal is asserted, the current bus cycle is terminated. If BERR is asserted before the falling edge of S4, AS will be negated in S7 in either a read or write cycle. As long as BERR remains asserted, the data and address buses will be in the high-impedance state. When BERR is negated, the processor will begin stacking for the exception sequence. Figure 21 is a timing diagram for the exception se-

Preliminary



R = Bus Request Internal
A = Bus Grant Acknowledge Internal
G = Bus Grant
T = Three-State Control to Bus Control Logic
X = Don't Care
* State machine will not change state if bus is in S0. Refer to BUS ARBITRATION CONTROL for additional information.

Figure 16. State Diagram of SCN68000 Bus Arbitration Unit



*This delay time is equal to parameter #33, t_{CHGL}

Figure 17. Timing Relationship of External Inputs to Internal Signals

quence which is composed of the following elements:

1. Stacking the program counter and status register.
2. Stacking the error information.
3. Reading the bus error vector table entry.
4. Executing the bus error handling routine.

The stacking of the program counter and the status register is the same as if an interrupt had occurred. Several additional items are stacked when a bus error occurs to determine the nature of the error and to correct it, if possible. The bus error vector is vector number two located at address $000008. The processor loads the program counter from this location and a software bus error handler routine is then executed by the processor. Refer to Exception Processing for additional information.

Rerunning the Bus Cycle — When the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the rerun sequence illustrated in figure 22.

The processor completes the bus cycle, then puts the address, data, function code, and control leads in the high-impedance state. The processor remains 'halted' and will not run another bus cycle until the halt signal is removed by external logic. The processor will then rerun the previous bus cycle using the same address, the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed before the halt signal is removed.

NOTE
The processor will not rerun a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a test-and-set operation is performed without ever releasing AS. If BERR and HALT are asserted during a read-modify-write cycle, a bus error operation results.

The processor terminates the bus cycle, then puts the address, data and function code output lines in the high-impedance state. The processor remains 'halted' and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will rerun the previous bus cycle using the same address, the same function codes, the same data (for a write operation), and the same controls. The bus error signal should be removed before the halt signal is removed.

Preliminary

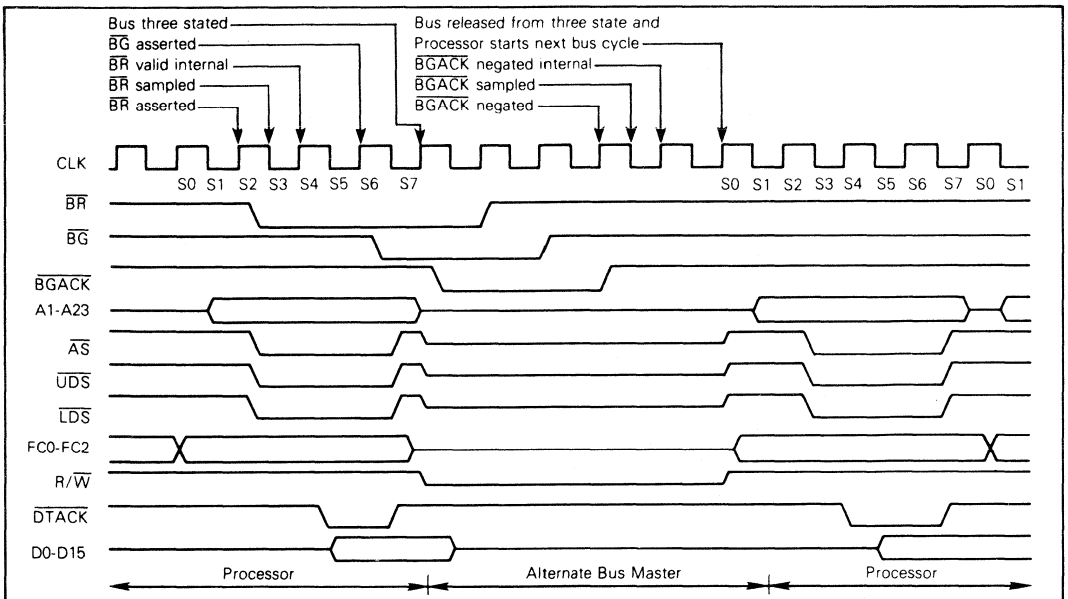


Figure 18. Bus Arbitration During Processor Bus Cycle

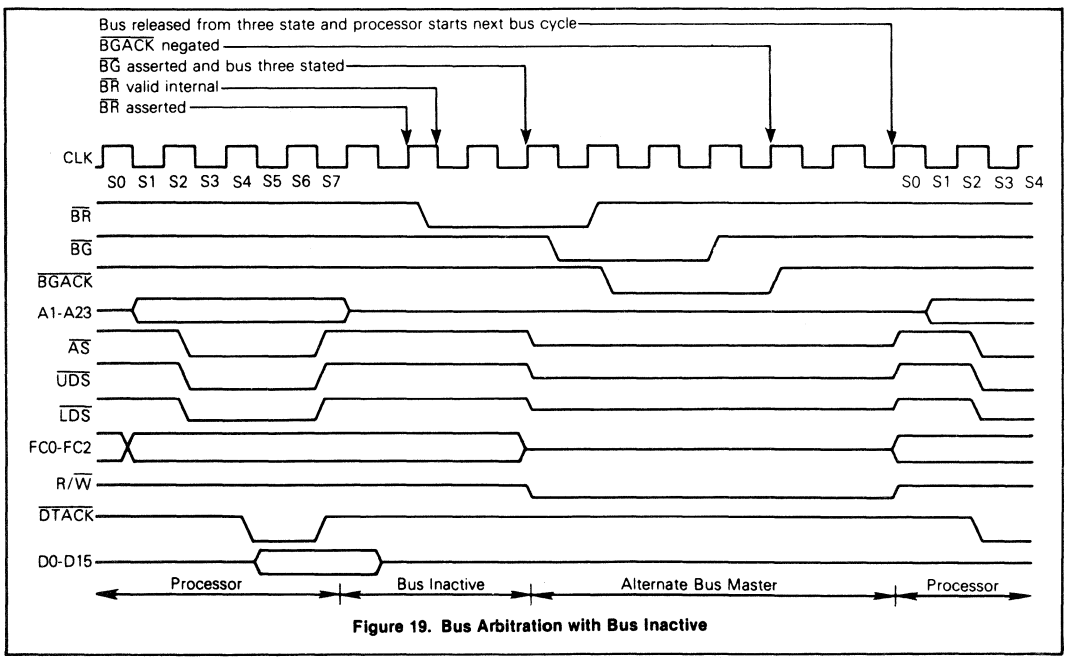


Figure 19. Bus Arbitration with Bus Inactive

Preliminary

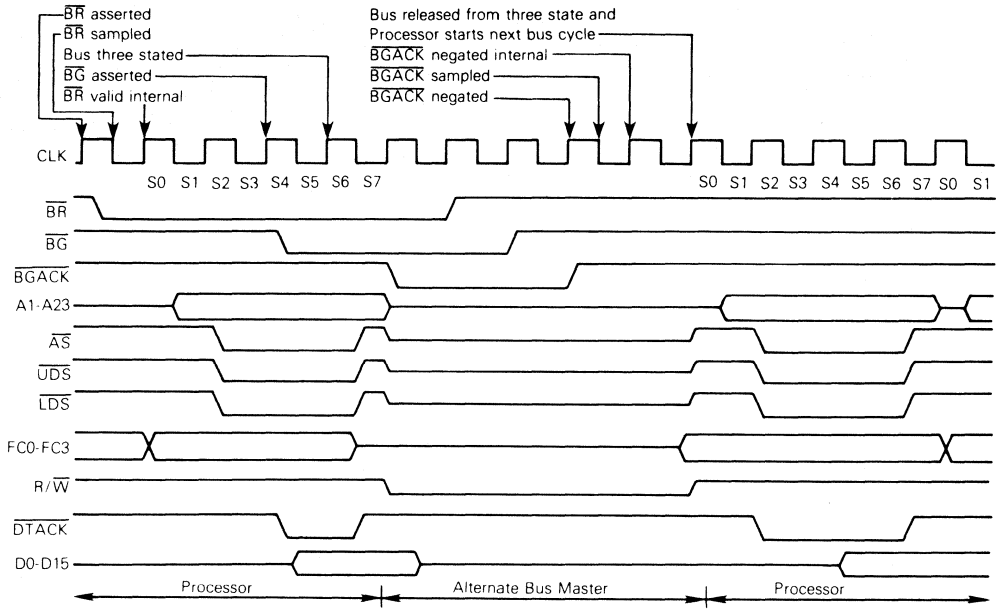


Figure 20. Bus Arbitration During Processor Bus Cycle Special Case

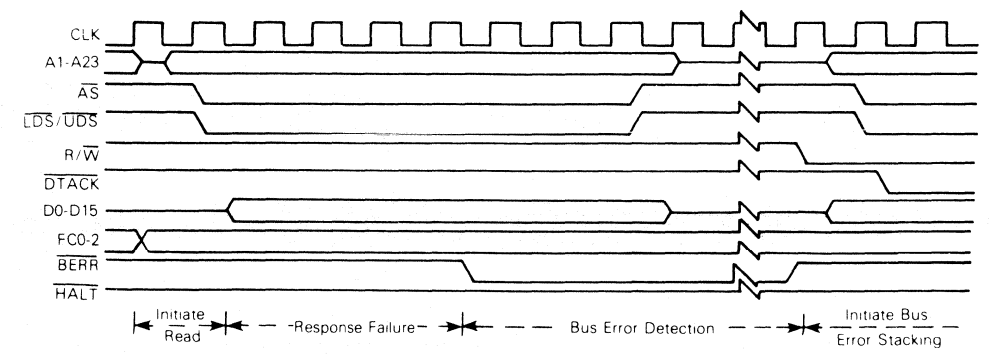


Figure 21. Bus Error Timing

Preliminary

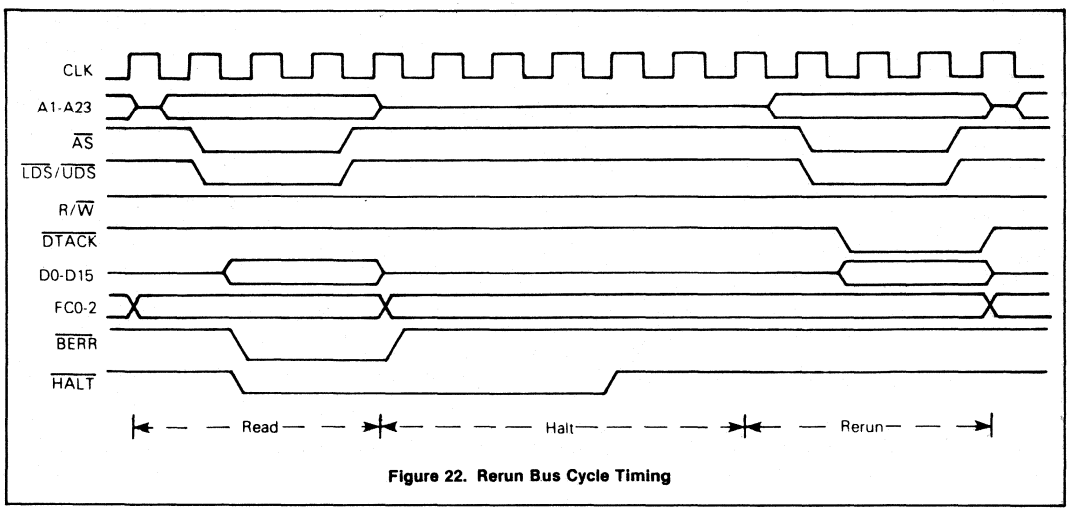


Figure 22. Rerun Bus Cycle Timing

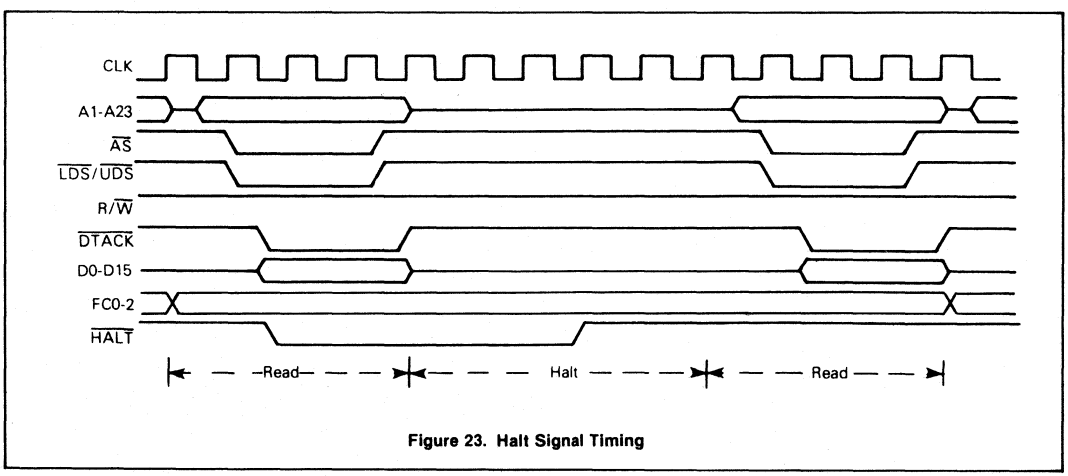


Figure 23. Halt Signal Timing

Halt Operation with No Bus Error — The halt input signal to the SCN68000 can be used to perform a halt/run/single-step function. The halt and run modes are somewhat self explanatory in that, when the halt signal is constantly active, the processor 'halts' (does nothing) and when the halt signal is constantly inactive, the processor 'runs' (does something).

The single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the 'run' mode until the processor starts a bus cycle, then changing to the 'halt' mode. Thus, the single step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

Figure 23 shows the timing required for correct single-step operations. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the processor.

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state. These include address lines and data lines. This is required for correct performance of the rerun bus cycle operation.

While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

Preliminary

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

Double Bus Faults — When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the processor. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row, which is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is rerun does not constitute a bus error exception, and does not contribute to a double bus fault. This means that as long as the external hardware requests it, the processor will continue to rerun the same bus cycle.

The bus error pin also has an effect on the processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

Relationship of DTACK, BERR, and HALT

In order to properly control termination of a bus cycle for a rerun or a bus error condition, DTACK, BERR, and HALT should be asserted and negated on the rising edge of the SCN68000 clock. This will assure that when two signals are asserted simultaneously, the required setup time (#47) for both of them will be met during the same bus state.

This, or some equivalent precaution, should be designed external to the SCN68000. Parameter #48 is intended to ensure this operation in a totally asynchronous system, and may be ignored if the above conditions are met.

The preferred bus cycle terminations can be summarized as follows (case numbers refer to table 4):

Normal termination: DTACK occurs first (case 1).

Halt termination: HALT is asserted at the same time, or precedes DTACK (no BERR) cases 2 and 3.

Bus error termination: BERR is asserted in lieu of, at same time, or preceding DTACK (case 4); BERR negated at same time, or after DTACK.

Rerun termination: HALT and BERR asserted at the same time, or before DTACK (cases 6 and 7); HALT must be negated at least one cycle after BERR. Case 5 indicates BERR can precede HALT which allows fully asynchronous assertion.

Table 4 details the resulting bus cycle termination under various combinations of control signal sequences. The negation of these same control signals under several conditions is shown in table 5 (DTACK is assumed to be negated normally in all cases; for best results, both DTACK and BERR should be negated when address strobe is negated).

Example A: A system uses a watch-dog timer to terminate accesses to unpopulated address space. The timer asserts DTACK and BERR simultaneous after time-out (case 4).

Table 4 DTACK, BERR, HALT ASSERTION RESULTS

Case No.	Control Signal	Asserted on Rising Edge of State N	Asserted on Rising Edge of State N+2	Result
1	$\overline{\text{DTACK}}$	A	S	Normal cycle terminate and continue.
	$\overline{\text{BERR}}$	NA	X	
	$\overline{\text{HALT}}$	NA	X	
2	$\overline{\text{DTACK}}$	A	S	Normal cycle terminate and halt. Continue when HALT removed.
	$\overline{\text{BERR}}$	NA	X	
	$\overline{\text{HALT}}$	A	S	
3	$\overline{\text{DTACK}}$	NA	A	Normal cycle terminate and halt. Continue when HALT removed.
	$\overline{\text{BERR}}$	NA	NA	
	$\overline{\text{HALT}}$	A	S	
4	$\overline{\text{DTACK}}$	X	X	Terminate and take bus error trap.
	$\overline{\text{BERR}}$	A	S	
	$\overline{\text{HALT}}$	NA	NA	
5	$\overline{\text{DTACK}}$	NA	X	Terminate and re-run.
	$\overline{\text{BERR}}$	A	S	
	$\overline{\text{HALT}}$	NA	A	
6	$\overline{\text{DTACK}}$	X	X	Terminate and re-run.
	$\overline{\text{BERR}}$	A	S	
	$\overline{\text{HALT}}$	A	S	
7	$\overline{\text{DTACK}}$	NA	X	Terminate and re-run when HALT removed.
	$\overline{\text{BERR}}$	NA	A	
	$\overline{\text{HALT}}$	A	S	

Legend:
N – the number of the current even bus state (e.g., S4, S6, etc.)
A – signal is asserted in this bus state
NA – signal is not asserted in this state
X – don't care
S – signal was asserted in previous state and remains asserted in this state

Preliminary

Example B: A system uses error detection on RAM contents. Designer can (a) delay DTACK until the data is verified, and return BERR and HALT simultaneously to rerun error cycle (case 6), or if valid, return DTACK; (b) delay DTACK until data is verified, and return BERR at the same time as DTACK if data is in error (case 4); (c) return DTACK prior to data verification, as described in the previous section. If data is invalid, BERR is asserted (case 1) in the next cycle. Error handling software must know how to recover the error cycle.

Reset Operation

The reset signal is a bidirectional signal that allows either the processor or an external signal to reset the system. Figure 24 illustrates reset timing. Both the halt and the reset lines must be applied to ensure total reset of the processor.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector number zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven. No other registers are affected by the reset sequence.

When a RESET instruction is executed, the processor drives the reset pin for 124 clock pulses. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor and its internal registers and the status register are unaffected. All external devices connected to the reset line should be reset at the completion of the RESET instruction.

Asserting the RESET and HALT pins for ten clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for 100 milliseconds.

Table 5 BERR AND HALT NEGATION RESULTS

Conditions of Termination in Table A	Control Signal	Negated on Rising Edge of State N		Negated on Rising Edge of State N+2	Results — Next Cycle
Bus Error	$\overline{BERR}$	●	or	●	Takes bus error trap.
	$\overline{HALT}$	●	or	●	
Re-run	$\overline{BERR}$	●	or	●	Illegal sequence; usually traps to vector number 0.
	$\overline{HALT}$	●			
Re-run	$\overline{BERR}$	●			Re-runs the bus cycle.
	$\overline{HALT}$			●	
Normal	$\overline{BERR}$	●			May lengthen next cycle.
	$\overline{HALT}$	●	or	●	
Normal	$\overline{BERR}$			●	If next cycle is started it will be terminated as a bus error.
	$\overline{HALT}$	●	or	none	

PROCESSING STATES

The SCN68000 is always in one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. A special case of the normal state is the stopped state which the processor enters when a STOP instruction is executed. In this state, no further memory references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor can handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

Privilege States

The processor operates in one of two states of privilege: the 'user' state or the 'supervisor' state. The privilege state determines which operations are legal, is used by the external memory management device to control and translate accesses,

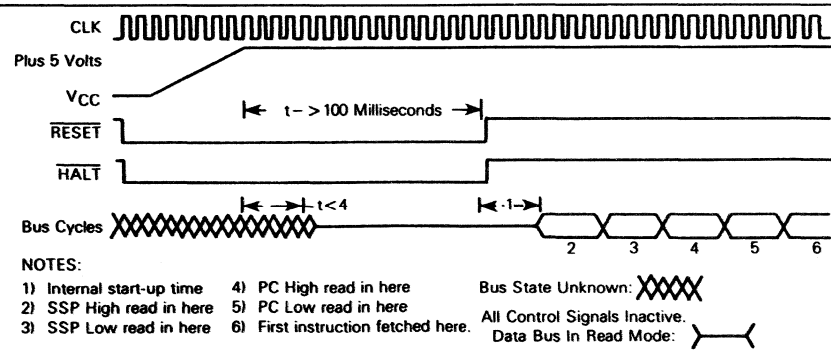


Figure 24. Reset Operation Timing

Preliminary

and is used to choose between the supervisor stack pointer and the user stack pointer in instruction references.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, their accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user state programs.

Supervisor State — The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S-bit of the status register: it is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the setting of the S-bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

User State — The user state is the lower state of privilege. For instruction execution, the user state is determined by the S-bit of the status register: it is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP instruction, or the RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move to user stack pointer (MOVE to USP) and move from user stack pointer (MOVE and USP) instructions are also privileged.

The bus cycles generated by an instruction executed in user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly, or address register seven explicity, access the user stack pointer.

Privilege State Changes — Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the current setting of the S-bit of the status register is saved and the S-bit is asserted, putting the processor in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

Reference Classification — When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor states, such as an interrupt acknowledge. Table 6 lists the classification of references.

Exception Processing General Description

The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made, and the status register is set for exception processing. In the second step the exception vector is determined, and the third step is the saving of the current processor context. In the fourth step a new context is obtained, and the processor switches to instruction processing.

Exception Vectors — Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (see figure 25), except for the reset vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an 8-bit number which, when multiplied by four, gives the address of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (see figure 26) to the processor on the data bus lines D0 through D7. The processor translates the vector number into a full 24-bit address, as shown in figure 27. The memory layout for exception vectors is given in table 7.

As shown in table 7, the memory layout is 512 words long (1024 bytes). It starts at address 0 and proceeds through address 1023. This provides 255 unique vectors; some of these are reserved for traps and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 64 entries, so the user interrupt vectors may overlap at the discretion of the systems designer.

Kinds of Exceptions — Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts and the bus error and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or

Table 6 REFERENCE CLASSIFICATION

Function Code Output			Reference Class
FC2	FC1	FC0	
0	0	0	(Unassigned)
0	0	1	User Data
0	1	0	User Program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor Data
1	1	0	Supervisor Program
1	1	1	Interrupt Acknowledge

Preliminary

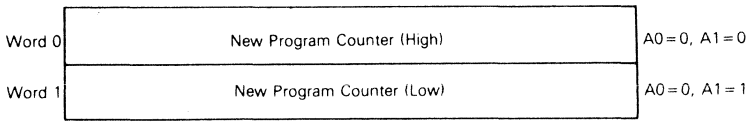


Figure 25. Exception Vector Format

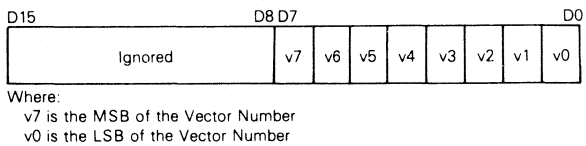


Figure 26. Peripheral Vector Number Format

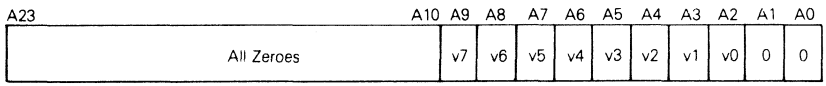


Figure 27. Address Translated from 8-Bit Vector Number

from address errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check register against bounds (CHK) and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word fetches from odd addresses and privilege violations cause exceptions. Tracing behaves like a very high priority, internally generated interrupt after each instruction execution.

Exception Processing Sequence — Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S-bit is asserted putting the processor into the supervisor privilege state. Also, the T-bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch, classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vectored number. This vector number is then used to generate the address of the exception vector.

The third step is to save the current processor status, except in the case of the reset exception. The current program counter value and the saved copy of the status register are stacked using the supervisor stack pointer. The program counter value stacked usually points to the next unexecuted instruction, however for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented from the address of the instruction which caused the error. Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector and the processor resumes instruction execution. The instruction at the address given in the exception vector is fetched, and the normal instruction decoding and execution is started.

Multiple Exceptions — The following describes the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted and the exception processing to commence at the next minor cycle of the processor. The group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but preempt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The group 2 exceptions occur as part of the

Preliminary

Table 7 EXCEPTION VECTOR ASSIGNMENT

Vector Number(s)	Address			Assignment
	Dec	Hex	Space	
0	0	000	SP	Reset: Initial SSP
–	4	004	SP	Reset: Initial PC
2	8	008	SD	Bus Error
3	12	00C	SD	Address Error
4	16	010	SD	Illegal Instruction
5	20	014	SD	Zero Divide
6	24	018	SD	CHK Instruction
7	28	01C	SD	TRAPV Instruction
8	32	020	SD	Privilege Violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 Emulator
11	44	02C	SD	Line 1111 Emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14*	56	038	SD	(Unassigned, reserved)
15	60	03C	SD	Uninitialized Interrupt Vector
16-23*	64	04C	SD	(Unassigned, reserved)
	95	05F		–
24	96	060	SD	Spurious Interrupt
25	100	064	SD	Level 1 Interrupt Autovector
26	104	068	SD	Level 2 Interrupt Autovector
27	108	06C	SD	Level 3 Interrupt Autovector
28	112	070	SD	Level 4 Interrupt Autovector
29	116	074	SD	Level 5 Interrupt Autovector
30	120	078	SD	Level 6 Interrupt Autovector
31	124	07C	SD	Level 7 Interrupt Autovector
32-47	128	080	SD	TRAP Instruction Vectors
	191	0BF		–
48-63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		–
64-255	256	100	SD	User Interrupt Vectors
	1023	3FF		–

*Vector numbers 12, 13, 14, 16 through 23 and 48 through 63 are reserved for future enhancements. No user peripheral devices should be assigned these numbers.

normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while group 2 exceptions have lowest priority. Within group 0, reset has highest priority, followed by bus error and then address error. Within group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is aborted. In another example, if an interrupt request occurs during the execution of an instruction while the T-bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in table 8.

Exception Processing Detailed Discussion

Exceptions have a number of sources, and each exception has processing which is peculiar to it.

Reset — The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, and the trace state is forced off. The processor interrupt priority mask is set at level seven. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, nstruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The RESET instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

Interrupts — Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, level seven being the highest priority. The status register contains a three-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero

Preliminary

Table 8 EXCEPTION GROUPING AND PRIORITY

Group	Exception	Processing
0	Reset Bus Error Address Error	Exception processing begins within two clock cycles.
1	Trace Interrupt Illegal Privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, Zero Divide	Exception processing is started by normal instruction execution

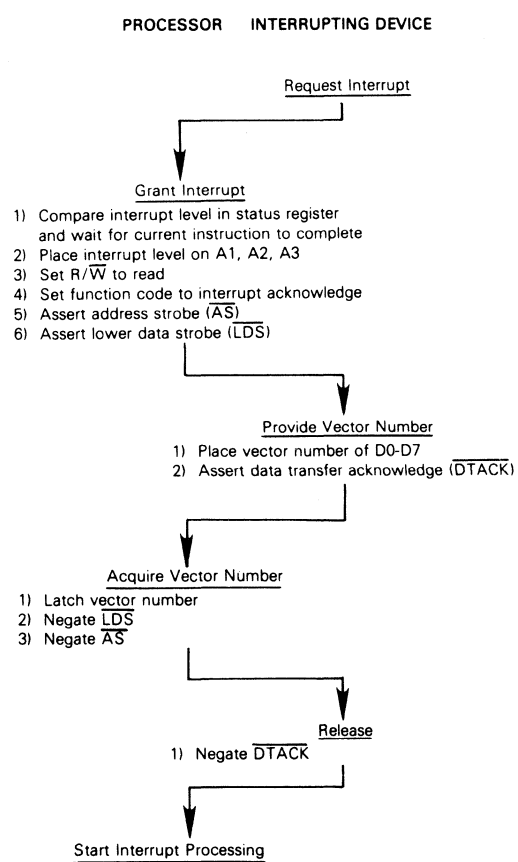


Figure 28. Interrupt Acknowledge Sequence Flow Chart

indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slighly different, as explained in a following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. First a copy of the status register is saved, and the privilege state is set to supervisor, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests an automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the program counter and status register on the supervisor stack. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flow chart for the interrupt acknowledge sequence is given in figure 28, a timing diagram is given in figure 29 and the interrupt exception timing sequence is shown in figure 30.

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a 'non-maskable interrupt' capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor is set to a lower level by an instruction.

Preliminary

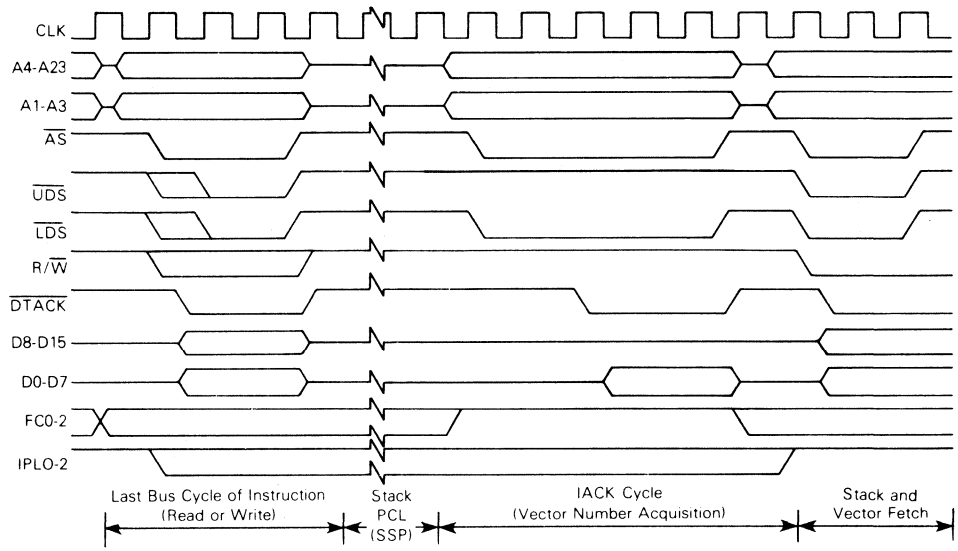


Figure 29. Interrupt Acknowledge Sequence Timing Diagram

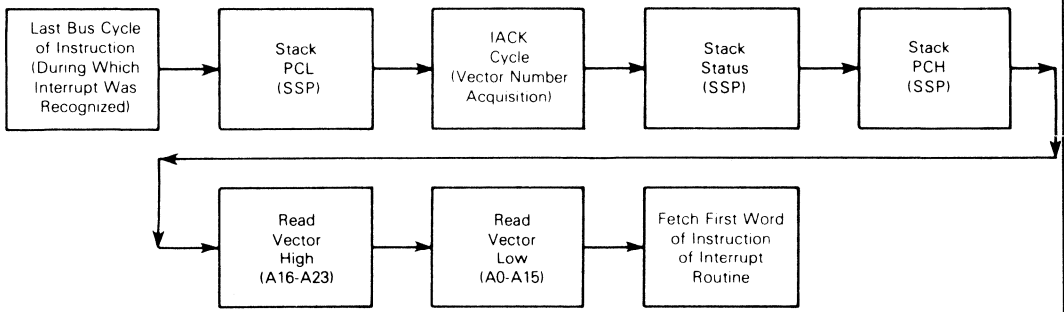


Figure 30. Interrupt Exception Timing Sequence

Preliminary

Uninitialized Interrupt — An interrupting device asserts VPA or provides an interrupt vector during an interrupt acknowledge cycle to the SCN68000. If the vector register has not been initialized, the responding SCN68000 family peripheral will provide vector 15, the uninitialized interrupt vector. This provides a uniform way to recover from a programming error.

Spurious Interrupt — If during the interrupt acknowledge cycle no device responds by asserting DTACK or VPA, the bus error line should be asserted to terminate the vector acquisition. The processor separates the processing of this error from the bus error by fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

Instruction Traps — Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception, and is useful for implementing system calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds. The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

Illegal and Unimplemented Instructions — Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit pattern of the first word of a legal instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs.

Word patterns with bits 15 through 12 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.

Privilege Violations — In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instructions are:

STOP
RESET
RTE
MOVE to SR
AND (word) immediate to SR
EOR (word) immediate to SR
OR (word) immediate to SR
MOVE USP

Tracing — To aid in program development, the SCN68000 includes a facility to allow instruction by instruction tracing. In the trace state, after each instruction is executed, an exception is forced allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T-bit in the supervisor portion of the status register. If the T-bit is negated (off), tracing is disabled and instruction execution proceeds from instruction to instruction as normal. If the T-bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated after the execution of that instruction is completed. If the instruction is not executed, either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction, an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

Bus Error — Bus error exceptions occur when external logic requests that a bus error be processed by an exception. The current bus cycle which the processor is making is then aborted. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing.

Exception processing for bus error follows the usual sequence of steps. The status register is copied, the supervisor state is entered, and the trace state is turned off. The vector number is generated to refer to the bus error vector. Since the processor was not between instructions when the bus error exception request was made, the context of the processor is more detailed. To save more of this context, additional information is saved on the supervisor stack. The program counter and the copy of the status register are of course saved. The value saved for the program counter is advanced by some amount, two to ten bytes beyond the address of the first word of the instruction which made the reference causing the bus error. If the bus error occurred during the fetch of next instruction, the saved program counter has a value in the vicinity of the current instruction, even if the current instruction is a branch, a jump, or a return instruction. Besides the usual information, the processor saves its internal copy of the first word of the instruction being processed, and the address which was being accessed by the aborted bus cycle. Specific information about the access is also saved: whether it was a read or a write, whether the processor was processing an instruction or not, and the classification displayed on the function code outputs when the bus error occurred. The processor is processing an instruction if it is in the normal state or processing a group 2 exception; the processor is not processing an instruction if it is processing a group 0 or group 1 exception. Figure 31 illustrates how this information is organized on the supervisor stack. Although this information is not sufficient in general to effect full recovery from the bus error, it does allow software diagnosis. Finally, the processor commences instruction processing at the address contained in the vector. It is the responsibility of the error handler routine to clean up the stack and determine where to continue execution.

If a bus error occurs during the exception processing for a bus error, address error, or reset, the processor is halted, and all processing ceases. This simplifies the detection of catastrophic system failure, since the processor removes itself from the system rather than destroying all memory contents. Only the RESET pin can restart a halted processor.

Address Error

Address error exceptions occur when the processor attempts to access a word or a long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor ceases whatever processing it is currently doing and begins exception processing.

Preliminary

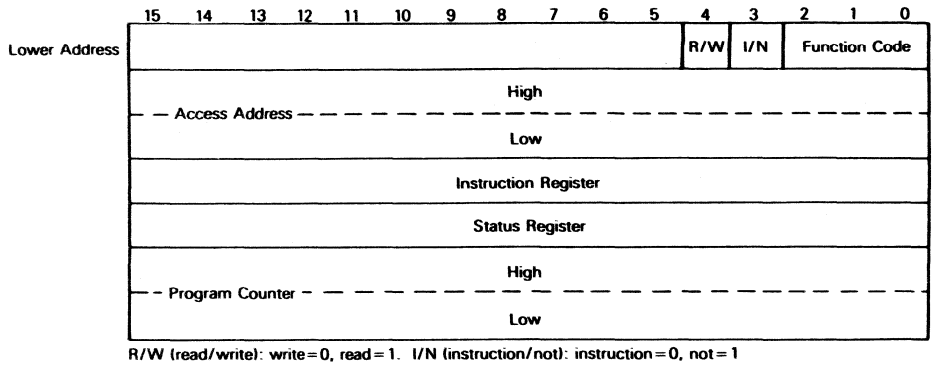


Figure 31. Supervisor Stack Order

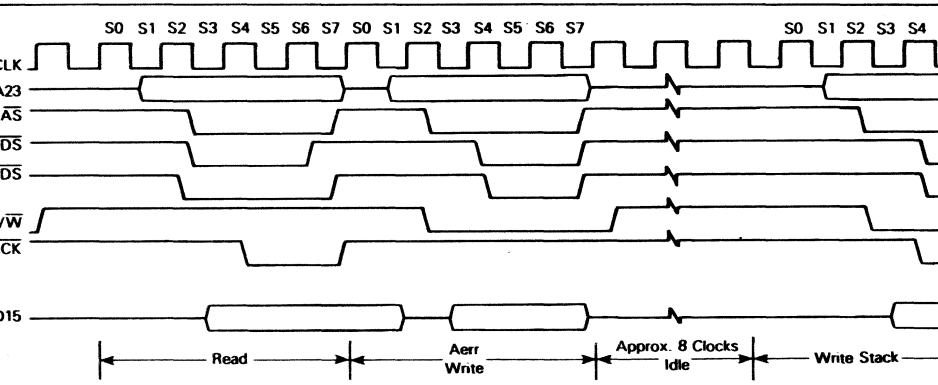


Figure 32. Address Error Timing

After exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector number refers to the address error vector instead. Likewise, if an address error occurs during the exception processing for a bus error, address error, or reset, the processor is halted. As show in figure 32, an address error will execute a short bus cycle followed by exception processing.

INTERFACE WITH SYNCHRONOUS PERIPHERALS

To interface synchronous peripherals with the asynchronous SCN68000, the processor modifies its bus cycle to meet the synchronous cycle requirements whenever a synchronous device address is detected. Figure 33 is a flow chart of the interface operation between the processor and synchronous devices.

Data Transfer Operation

Three signals on the processor provide the synchronous interface. They are: enable (E), valid memory address (VMA), and valid peripheral address (VPA). The bus frequency is one tenth of the incoming SCN68000 clock frequency. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive VPA accesses on successive E pulses.

The synchronous cycle timing diagrams and corresponding AC electrical characteristics table are located towards the end of this data sheet. At state zero (S0) in the cycle, the address bus and function codes are in the high-impedance state. One half clock later, in state 1, the address bus and function code outputs are released from the high-impedance state.

During state 2, the address strobe (AS) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle, the read/write (R/W) signal is switched to low (write) during state 2. One half clock later, in state 3, the write data is placed on the

Preliminary

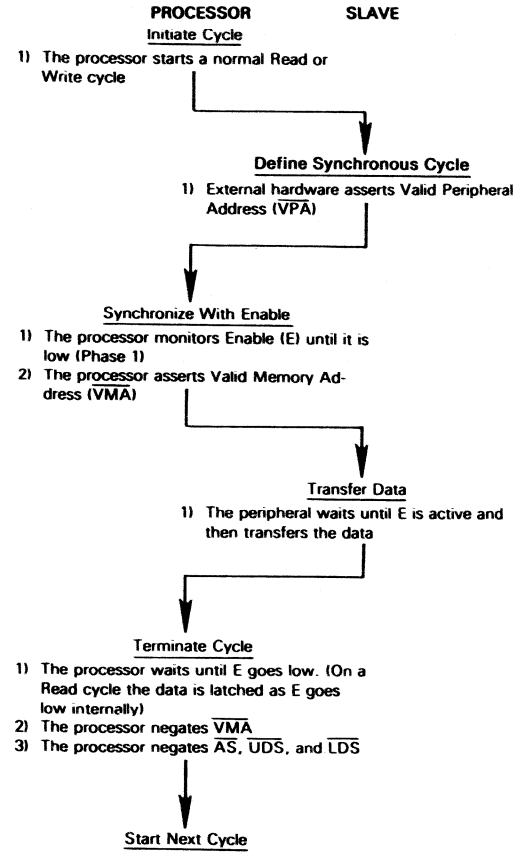


Figure 33. Synchronous Interfacing Flow Chart

data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus.

The processor now inserts wait states until it recognizes the assertion of VPA. The VPA input signals the processor that the address on the bus is the address of a synchronous device (or an area reserved for synchronous devices) and that the bus should conform to the synchronous transfer characteristics of the synchronous bus. Valid peripheral address is derived by decoding the address bus, conditioned by the address strobe.

After the recognition of VPA, the processor assures that enable (E) is low, by waiting if necessary, and subsequently asserts VMA. Valid memory address is then used as part of the chip select equation of the peripheral. This ensures that the synchronous peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Cycle timing diagrams depicting best and worst cases are located towards the end of this data sheet. The cycle length is dependent strictly on when VPA is asserted in relationship to the E clock.

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one half clock cycle later in state 7, and the enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state and the read/write signal is switched high at this time. The peripheral logic must remove VPA within one clock after address strobe is negated.

DTACK should not be asserted while VPA is asserted. The SCN68000 VMA is active low, while the VMA of the synchronous device should be active high. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting peripherals.

Interrupt Operation

During an interrupt acknowledge cycle while the processor is fetching the vector, if VPA is asserted, the SCN68000 will assert VMA and complete a synchronous read cycle as shown in figure 34. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).

There are six normal interrupt vectors and one NMI type vector. As with the SCN68000's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since VMA is asserted during autovectoring, the synchronous peripheral address decoding should prevent unintended accesses.

DATA TYPES AND ADDRESSING MODES

Five basic data types are supported:

- Bits
- BCD digits (8-bits)
- Bytes (8-bits)
- Word (16-bits)
- Long words (32-bits)

In addition, operations on other data types such as memory addresses, status word data, etc. are provided for in the instruction set. The 14 addressing modes (see table 9) include six basic types:

Preliminary

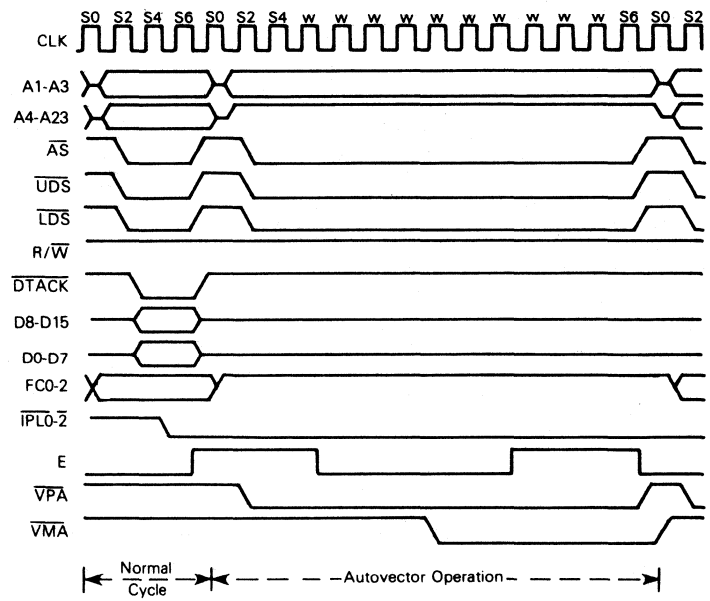


Figure 34. Autovector Operation Timing

- Register direct
- Register indirect
- Absolute
- Immediate
- Program counter relative
- Implied

Included in the register indirect addressing modes is the capability to do post-incrementing, pre-decrementing, offsetting and indexing. Program counter relative mode can also be modified via indexing and offsetting.

Instruction Format

Instructions are from one to five words in length, a shown in figure 35. The length of the instruction and the operation to be performed are specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address modes specified in the operation word.

Program/Data References

The SCN68000 separates memory references into two classes: program references and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Generally, operand reads are from the data space. All operand writes are to the data space.

Addressing

Instructions for the SCN68000 contain two kinds of information; the type of function to be performed and the location of the operand(s) on which to perform the function. Instructions specify an operand location in one of three ways:

Register specification — the number of the register is given in the register field of the instruction.

Effective address — use of the different effective address modes.

Implicit reference — the definition of certain instructions implies the use of specific registers.

Register Specification

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

Effective Address

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, figure 36 shows the general format of the single effective address instruction operation word. The effective address is composed of two 3-bit fields: the mode field, and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

Preliminary

Table 9 DATA ADDRESSING MODES

Mode	Generation
Register Direct Addressing	
Data Register Direct	EA = Dn
Address Register Direct	EA = An
Absolute Data Addressing	
Absolute Short	EA = (Next Word)
Absolute Long	EA = (Next Two Words)
Program Counter Relative Addressing	
Relative with Offset	EA = (PC) + d_{16}
Relative with Index and Offset	EA = (PC) + (Xn) + d_8
Register Indirect Addressing	
Register Indirect	EA = (An)
Postincrement Register Indirect	EA = (An), An ← An + N
Predecrement Register Indirect	An ← An − N, EA = (An)
Register Indirect with Offset	EA = (An) + d_{16}
Indexed Register Indirect with Offset	EA = (An) + (Xn) + d_8
Immediate Data Addressing	
Immediate	DATA = Next Word(s)
Quick Immediate	Inherent Data
Implied Addressing	
Implied Register	EA = SR, USP, SP, PC

NOTES:

EA = Effective Address
An = Address Register
Dn = Data Register
Xn = Address or Data Register used as Index Register
SR = Status Register
PC = Program Counter
( ) = Contents of
d_8 = Eight-bit Offset (displacement)
d_{16} = Sixteen-bit Offset (displacement)
N = 1 for Byte, 2 for Words and 4 for Long Words
← = Replaces

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in a following word or words and is considered part of the instruction, as shown in figure 35. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

Register Direct Modes

These effective addressing modes specify that the operand is in one of the 16 multifunction registers.

Data Register Direct — The operand is in the data register specified by the effective address register field.

Address Register Direct — The operand is in the address register specified by the effective address register field.

Memory Address Modes

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

Address Register Indirect — The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

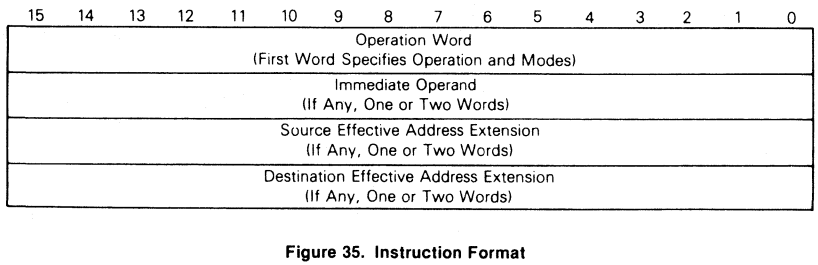


Figure 35. Instruction Format

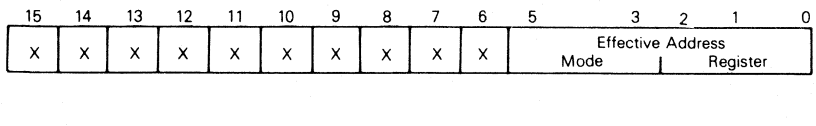


Figure 36. Single Effective Address Instruction Operation Word General Format

Preliminary

Address Register Indirect with Postincrement — The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending on whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect with Predecrement — The address of the operand is in the address register specified by the register field. Before the operand and the address is used, it is decremented by one, two or four depending on whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect with Displacement — This address mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Address Register Indirect with Index — This address mode requires one word of extension. The address of the operand is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Special Addressing Modes

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address — This address mode requires one word of extension. The address of the operand is in the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump to subroutine instructions.

Absolute Long Address — This address mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high-order part of the address is the first extension word; the low order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and the jump to subroutine instructions.

Program Counter with Displacement — This address mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Program Counter with Index — This address mode requires one word of extension. The address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The value in the program counter is the address of the extension word. This reference is classifed as a program reference.

Immediate Data — This address mode requires either one or two words of extension depending on the size of the operation.

Byte operation — operand is low order byte of extension word.

Word operation — operand is extension word.

Long word operation — operand is in the two extension words, high-order 16 bits are in the first extension word, low-order 16 bits are in the second extension word.

Condition Codes or Status Register — A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR
ANDI to SR
EORI to SR
ORI to CCR
ORI to SR

Effective Address Encoding Summary

Table 10 is a summary of the effective addressing modes.

Implicit Reference

Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), or the status register (SR).

System Stack

The system stack is used implicitly by many instructions; user stacks and

Table 10 EFFECTIVE ADDRESS ENCODING SUMMARY

Addressing Mode	Mode	Register
Data Register Direct	000	register number
Address Register Direct	001	register number
Address Register Indirect	010	register number
Address Register Indirect with Postincrement	011	register number
Address Register Indirect with Predecrement	100	register number
Address Register Indirect with Displacement	101	register number
Address Register Indirect with Index	110	register number
Absolute Short	111	000
Absolute Long	111	001
Program Counter with Displacement	111	010
Program Counter with Index	111	011
Immediate	111	100

Preliminary

queues may be created and maintained through the addressing modes. Address register seven (A7) is the stack pointer (SP). The SP is either the SSP or the USP, depending on the state of the S-bit in the status register. If the S-bit indicates supervisor state, SSP is the active system stack pointer, and the USP cannot be referenced as an address register. If the S-bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

INSTRUCTION SET OVERVIEW

The SCN68000 instruction set is summarized in table 11. Some additional instructions are variations, or subsets, of these and appear in table 12. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words, and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, 'quick' arithmetic operations, BCD arithmetic and expanded operations (through traps).

The instructions form a set of tools that include all machine functions to perform the following operations:

Data movement
Integer arithmetic

Table 11 INSTRUCTION SET SUMMARY

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{CC}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{CC}	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide

Mnemonic	Description
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or

Mnemonic	Description
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{CC}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

Table 12 VARIATIONS OF INSTRUCTION TYPES

Instruction Type	Variation	Description
ADD	**ADD**	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	**AND**	Logical And
	ANDI	And Immediate
CMP	**CMP**	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	**EOR**	Exclusive Or
	EORI	Exclusive Or Immediate

Instruction Type	Variation	Description
MOVE	**MOVE**	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	**NEG**	Negate
	NEGX	Negate with Extend
OR	**OR**	Logical Or
	ORI	Or Immediate
SUB	**SUB**	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

Preliminary

Logical
Shift and rotate
Bit manipulation
Binary coded decimal
Program control
System control

The complete range of instruction capabilities, combined with the variety of addressing modes described previously, provide a very flexible base for program development.

Data Movement Operations

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data move instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to register. Address move instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction, there are several special data movement instructions: move multiple register (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), and move quick (MOVEQ). Table 13 is a summary of the data movement operations.

Integer Arithmetic Operations

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions can be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotient with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extended (EXT), and negate binary with extend (NEGX).

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 14 is a summary of the integer arithmetic operations.

Table 13 DATA MOVEMENT OPERATIONS

Instruction	Operand Size	Operation
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	An → SP@ – SP → An SP + d → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVEP	16, 32	(EA) → Dn Dn → EA
MOVEQ	8	#xxx → Dn
PEA	32	EA → SP@ –
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → Sp SP@ + → An

NOTES:
s = source
d = destination
[ ] = bit numbers
@ – = indirect with predecrement
@ + = indirect with postdecrement

Logical Operations

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 15 is a summary of the logical operations.

Shift and Rotate Operations

Shift operations in both directions are provided by the arithmetic instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in the instruction of one to eight bits, or 0 to 63 specified in a data register. Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates. Table 16 is a summary of the shift and rotate operations.

Bit Manipulation Operations

Bit manipulation operations are accomplished using bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 17 is a summary of the bit manipulation operations (bit 2 of the status register is Z).

Binary Coded Decimal Operations

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 18 is a summary of the binary coded decimal operations.

Program Control Operations

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions (see table 19). The conditional instructions provide setting and branching for the following conditions:

CC—carry clear
CS—carry set
EQ—equal
F—never true
GE—greater or equal
GT—greater than
HI—high
LE—less or equal
LS—low or same

Preliminary

LT—less than
MI—minus
NE—not equal
PL—plus
T—always true
VC—no overflow
VS—overflow

System Control Operations

System control operations are accomplished by using priviledged instructions, trap generating instructions, and instructions that use or modify the status register. These instructions are summarized in table 20.

INSTRUCTION SET

The following provides information about the addressing categories and instruction set of the SCN68000.

Addressing Categories

Effective address modes can be categorized by the ways in which they may be

Table 14 INTEGER ARITHMETIC OPERATIONS

Instruction	Operand Size	Operation
ADD	8, 16, 32	Dn + (EA) → Dn
		(EA) + Dn → EA
		(EA) + #xxx → EA
	16, 32	An + (EA) → An
ADDX	8, 16, 32	Dx + Dy + X → Dx
	16, 32	Ax@ – Ay@ – + X → Ax@
CLR	8, 16, 32	0 → EA
CMP	8, 16, 32	Dn – (EA)
		(EA) – #xxx
		Ax@ + – Ay@ +
	16, 32	An – (EA)
DIVS	32 ÷ 16	Dn/(EA) → Dn
DIVU	32 ÷ 16	Dn/(EA) → Dn
EXT	8 → 16	$(Dn)_8 \rightarrow Dn_{16}$
	16 → 32	$(Dn)_{16} \rightarrow Dn_{32}$
MULS	16*16 → 32	Dn*(EA) → Dn
MULU	16*16 → 32	Dn*(EA) → Dn
NEG	8, 16, 32	0 – (EA) → EA
NEGX	8, 16, 32	0 – (EA) – X – EA
SUB	8, 16, 32	Dn – (EA) → Dn
		(EA) – Dn → EA
		(EA) – #xxx → EA
	16, 32	An – (EA) → An
SUBX	8, 16, 32	Dx – Dy – X → Dx
		Ax@ – – Ay@ – – X → Ax@
TAS	8	(EA) – 0, 1 → EA[7]
TST	8, 16, 32	(EA) – 0

NOTE: [ ] = bit number

Table 15 LOGICAL OPERATIONS

Instruction	Operand Size	Operation
AND	8, 16, 32	DnΛ(EA) → Dn
		(EA)ΛDn → EA
		(EA)Λ#xxx → EA
OR	8, 16, 32	Dn v (EA) → Dn
		(EA) v Dn → EA
		(EA) v #xxx → EA
EOR	8, 16, 32	(EA) ⊕ Dy → EA
		(EA) ⊕ #xxx → EA
NOT	8, 16, 32	~(EA) → EA

NOTE: ~ = invert

Table 16 SHIFT AND ROTATE OPERATIONS

Instruction	Operand Size	Operation
ASL	8, 16, 32	X/C ← ← ← 0
ASR	8, 16, 32	→ → → X/C
LSL	8, 16, 32	X/C ← ← ← 0
LSR	8, 16, 32	0 → → → X/C
ROL	8, 16, 32	C ← ← ←
ROR	8, 16, 32	→ → → C
ROXL	8, 16, 32	C ← ← ← X
ROXR	8, 16, 32	X → → → C

Table 17 BIT MANIPULATION OPERATIONS

Instruction	Operand Size	Operation
BTST	8, 32	~bit of (EA) → Z
BSET	8, 32	~bit of (EA) → Z
		1 → bit of EA
BCLR	8, 32	~bit of (EA) → Z
		0 → bit of EA
BCHG	8, 32	~bit of (EA) → Z
		~bit of (EA) → bit of EA

Table 18 BINARY CODED DECIMAL OPERATIONS

Instruction	Operand Size	Operation
ABCD	8	$Dx_{10} + Dy_{10} + X \rightarrow Dx$
		$Ax@_{-10} + Ay@_{-10} + X \rightarrow Ax@$
SBCD	8	$Dx_{10} - Dy_{10} - X \rightarrow Dx$
		$Ax@_{-10} - Ay@_{-10} - X \rightarrow Ax@$
NBCD	8	$0 - (EA)_{10} - X \rightarrow EA$

Preliminary

used. The following classifications are used in the instruction definitions:

Data — If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

Memory — If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable — If an effective address mode may be used to refer to alterable (writable) operands, it is considered an alterable addressing effective address mode.

Control — If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 21 shows the various categories to which each of the effective address modes belongs. Table 22 shows the instruction set.

The status register addressing mode is not permitted unless it is explicitly mentioned as a legal addressing mode.

These categories may be combined, so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable.

Instruction Prefetch

The SCN68000 uses a two word tightly coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1. When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2. In the case of multiword instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3. The last fetch from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4. If the instruction is a single word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch. In the case of an interrupt or trace exception, both words are not used.
5. The program counter usually points to the last word fetched from the instruction stream.

INSTRUCTION EXECUTION TIMES

The following contains listings of the instruction execution times in terms of the external clock (CLK) periods. In this timing data, it is also assumed that the memory cycle time is no greater than four periods

Table 19 PROGRAM CONTROL OPERATIONS

Instruction	Operation
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8- and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTR	Return and restore condition codes
RTS	Return from subroutine

Table 20 SYSTEM CONTROL OPERATIONS

Instruction	Operation
Privileged	
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
ORI to SR	Logical OR to status register
MOVE USP	Move user stack pointer
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
Trap Generating	
TRAP	Trap
TRAPV	Trap on overflow
CHK	Check register against bounds
Status Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
ORI to CCR	Logical OR to condition codes
MOVE SR to EA	Store status register

Preliminary

Table 21 EFFECTIVE ADDRESSING MODE CATEGORIES

Effective Address Modes	Mode	Register	Data	Addressing Categories		
				Memory	Control	Alterable
Dn	000	register number	X	–	–	X
An	001	register number	–	–	–	X
An@	010	register number	X	X	X	X
An@+	011	register number	X	X	–	X
An@–	100	register number	X	X	–	X
An@(d)	101	register number	X	X	X	X
An@(d, ix)	110	register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
PC@(d)	111	010	X	X	X	–
PC@(d, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

of the external processor clock input, which prevents the insertion of wait states in the bus cycle. The number of bus read and write cycles for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as (r/w), where r is the number of read cycles and w is the number of write cycles. The number of periods includes instruction fetch and all applicable operand fetches and stores.

Effective Address Operand Calculation Timing

Table 23 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching the memory operand. The number of bus read and write cycles is shown in parenthesis as (r/w). Note that there are no write cycles involved in processing the effective address.

Move Instruction Clock Periods

Table 24 and 25 indicate the number of clock periods for the move instruction. This data includes fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as (r/w).

Standard Instruction Clock Periods

The number of clock periods shown in table 26 indicates the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read cycles must be added to those of the effective address calculated where indicated.

In table 26, the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

Immediate Instruction Clock Periods

The number of clock periods shown in table 27 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

In table 27, the headings have the following meanings: # = immediate operand, Dn = data register operand, M = memory operand, and SR = status register.

Single Operand Instruction Clock Periods

Table 28 indicates the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Shift/Rotate Instruction Clock Periods

Table 29 indicates the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Bit Manipulation Instruction Clock Periods

Table 30 indicates the number of clock periods for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Conditional Instruction Clock Periods

Table 31 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

JMP, JSR, LEA, PEA, MOVEM Instruction Clock Periods

Table 32 indicates the number of clock periods required for the jump, jump to subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as (r/w).

Preliminary

Table 22 INSTRUCTION SET

Mnemonic	Description	Operation	Condition Codes				
			X	N	Z	V	C
ABCD	Add Decimal with Extend	(Destination)$_{10}$ + (Source)$_{10}$ → Destination	•	U	•	U	•
ADD	Add Binary	(Destination) + (Source) → Destination	•	•	•	•	•
ADDA	Add Address	(Destination) + (Source) → Destination	–	–	–	–	–
ADDI	Add Immediate	(Destination) + Immediate Data → Destination	•	•	•	•	•
ADDQ	Add Quick	(Destination) + Immediate Data → Destination	•	•	•	•	•
ADDX	Add Extended	(Destination) + (Source) + X → Destination	•	•	•	•	•
AND	AND Logical	(Destination) ∧ (Source) → Destination	–	•	•	0	0
ANDI	AND Immediate	(Destination) ∧ Immediate Data → Destination	–	•	•	0	0
ASL, ASR	Arithmetic Shift	(Destination) Shifted by <count> → Destination	•	•	•	•	•
B$_{CC}$	Branch Conditionally	If $_{CC}$ then PC + d → PC	–	–	–	–	–
BCHG	Test a Bit and Change	~(<bit number>) OF Destination → Z ~(<bit number>) OF Destination → <bit number> OF Destination	–	–	•	–	–
BCLR	Test a Bit and Clear	~(<bit number>) OF Destination → Z 0 → <bit number> → OF Destination	–	–	•	–	–
BRA	Branch Always	PC + d → PC	–	–	–	–	–
BSET	Test a Bit and Set	~(<bit number>) OF Destination → Z 1 → <bit number> OF Destination	–	–	•	–	–
BSR	Branch to Subroutine	PC → SP@ – ; PC + d → PC	–	–	–	–	–
BTST	Test a Bit	~(<bit number>) OF Destination → Z	–	–	•	–	–
CHK	Check Register against Bounds	If Dn <0 or Dn> (<ea>) then TRAP	–	•	U	U	U
CLR	Clear an Operand	0 → Destination	–	0	1	0	0
CMP	Compare	(Destination) – (Source)	–	•	•	•	•
CMPA	Compare Address	(Destination) – (Source)	–	•	•	•	•
CMPI	Compare Immediate	(Destination) – Immediate Data	–	•	•	•	•
CMPM	Compare Memory	(Destination) – (Source)	–	•	•	•	•
DB$_{CC}$	Test Condition, Decrement and Branch	If ~ $_{CC}$ then Dn – 1 → Dn; if Dn ≠ – 1 then PC + d → PC	–	–	–	–	–
DIVS	Signed Divide	(Destination)/(Source) → Destination	–	•	•	•	0
DIVU	Unsigned Divide	(Destination)/(Source) → Destination	–	•	•	•	0
EOR	Exclusive OR Logical	(Destination) ⊕ (Source) → Destination	–	•	•	0	0
EORI	Exclusive OR Immediate	(Destination) ⊕ Immediate Data → Destination	–	•	•	0	0
EXG	Exchange Register	Rx ↔ Ry	–	–	–	–	–
EXT	Sign Extend	(Destination) Sign-extended → Destination	–	•	•	0	0
JMP	Jump	Destination → PC	–	–	–	–	–
JSR	Jump to Subroutine	PC → SP@ – ; Destination → PC	–	–	–	–	–
LEA	Load Effective Address	Destination → An	–	–	–	–	–
LINK	Link and Allocate	An → SP@ – ; SP → An; SP + d → SP	–	–	–	–	–
LSL, LSR	Logical Shift	(Destination) Shifted by <count> → Destination	•	•	•	0	•
MOVE	Move Data from Source to Destination	(Source) → Destination	–	•	•	0	0
MOVE to CCR	Move to Condition Code	(Source) → CCR	•	•	•	•	•
MOVE to SR	Move to the Status Register	(Source) → SR	•	•	•	•	•

• affected 0 cleared U defined
– unaffected 1 set

Preliminary

Table 22 INSTRUCTION SET (Continued)

Mnemonic	Description	Operation	Condition Codes				
			X	N	Z	V	C
MOVE from SR	Move from the Status Register	SR → Destination	–	–	–	–	–
MOVE USP	Move User Stack Poihter	USP → An; An → USP	–	–	–	–	–
MOVEA	Move Address	(Source) → Destination	–	–	–	–	–
MOVEM	Move Multiple Registers	Registers → Destination (Source) → Registers	–	–	–	–	–
MOVEP	Move Peripheral Data	(Source) → Destination	–	–	–	–	–
MOVEQ	Move Quick	Immediate Data → Destination	–	•	•	0	0
MULS	Signed Multiply	(Destination)*(Source) → Destination	–	•	•	0	0
MULU	Unsigned Multiply	(Destination)*(Source) → Destination	–	•	•	0	0
NBCD	Negate Decimal with Extend	$0-(\text{Destination})_{10}-X \rightarrow$ Destination	•	U	•	U	•
NEG	Negate	0 – (Destination) → Destination	•	•	•	•	•
NEGX	Negate with Extend	0 – (Destination) – X → Destination	•	•	•	•	•
NOP	No Operation	–	–	–	–	–	–
NOT	Logical Complement	~(Destination) → Destination	–	•	•	0	0
OR	Inclusive OR Logical	(Destination) v (Source) → Destination	–	•	•	0	0
ORI	Inclusive OR Immediate	(Destination) v Immediate Data → Destination	–	•	•	0	0
PEA	Push Effective Address	Destination → SP@ –	–	–	–	–	–
RESET	Reset External Devices	–	–	–	–	–	–
ROL, ROR	Rotate (Without Extend)	(Destination) Rotated by <count> → Destination	–	•	•	0	•
ROXL, ROXR	Rotate with Extend	(Destination) Rotated by <count> → Destination	•	•	•	0	•
RTE	Return from Exception	SP@ + → SR; SP@ + → PC	•	•	•	•	•
RTR	Return and Restore Condition Codes	SP@ + → CC; SP@ + → PC	•	•	•	•	•
RTS	Return from Subroutine	SP@ + → PC	–	–	–	–	–
SBCD	Subtract Decimal with Extend	$(\text{Destination})_{10}-(\text{Source})_{10}-X \rightarrow$ Destination	•	U	•	U	•
S_{CC}	Set According to Condition	If CC then 1's → Destination else 0's → Destination	–	–	–	–	–
STOP	Load Status Register and Stop	Immediate Data → SR; STOP	•	•	•	•	•
SUB	Subtract Binary	(Destination) – (Source) → Destination	•	•	•	•	•
SUBA	Subtract Address	(Destination) – (Source) → Destination	–	–	–	–	–
SUBI	Subtract Immediate	(Destination) – Immediate Data → Destination	•	•	•	•	•
SUBQ	Subtract Quick	(Destination) – Immediate Data → Destination	•	•	•	•	•
SUBX	Subtract with Extend	(Destination) – (Source) – X → Destination	•	•	•	•	•
SWAP	Swap Register Halves	Register [31:16] ↔ Register [15:0]	–	•	•	0	0
TAS	Test and Set an Operand	(Destination) Tested → CC; 1 → [7] OF Destination	–	•	•	0	0
TRAP	Trap	PC → SSP@ – ; SR → SSP@ – ; (Vector) → PC	–	–	–	–	–
TRAPV	Trap on Overflow	If V then TRAP	–	–	–	–	–
TST	Test an Operand	(Destination) Tested → CC	–	•	•	0	0
UNLK	Unlink	An → SP; SP@ + → An	–	–	–	–	–

[ ] = bit number

• affected 0 cleared U defined
– unaffected 1 set

Preliminary

Table 23 EFFECTIVE ADDRESS OPERAND CALCULATION TIMING

Addressing Mode		Byte, Word	Long
	Register		
Dn	Data Register Direct	0(0/0)	0(0/0)
An	Address Register Direct	0(0/0)	0(0/0)
	Memory		
An@	Address Register Indirect	4(1/0)	8(2/0)
An@+	Address Register Indirect with Postincrement	4(1/0)	8(2/0)
An@–	Address Register Indirect with Predecrement	6(1/0)	10(2/0)
An@(d)	Address Register Indirect with Displacement	8(2/0)	12(3/0)
An@(d, ix)*	Address Register Indirect with Index	10(2/0)	14(3/0)
xxx.W	Absolute Short	8(2/0)	12(3/0)
xxx.L	Absolute Long	12(3/0)	16(4/0)
PC@(d)	Program Counter with Displacement	8(2/0)	12(3/0)
PC@(d, ix)*	Program Counter with Index	10(2/0)	14(3/0)
#xxx	Immediate	4(1/0)	8(2/0)

*The size of the index register (ix) does not affect execution time.

Table 24 MOVE BYTE AND WORD INSTRUCTION CLOCK PERIODS

Source	Destination								
	Dn	An	An@	An@+	An@–	An@(d)	An@(d,ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An@	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@+	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
An@–	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
An@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
An@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx.W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx.L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
PC@(d)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
PC@(d, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

The size of the index register (ix) does not affect execution time.

Table 25 MOVE LONG INSTRUCTION CLOCK PERIODS

Source	Destination								
	Dn	An	An@	An@+	An@–	An@(d)	An@(d,ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An@	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@+	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
An@–	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
An@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
An@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx.W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx.L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
PC@(d)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(5/2)
PC@(d, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

*The size of the index register (ix) does not affect execution time.

Preliminary

Table 26 STANDARD INSTRUCTION CLOCK PERIODS

Instruction	Size	op <ea>, An	op <ea>, Dn	op Dn, <M>
ADD	Byte, Word	8(1/0) +	4(1/0) +	8(1/1) +
	Long	6(1/0) + **	6(1/0) + **	12(1/2) +
AND	Byte, Word	–	4(1/0) +	8(1/1) +
	Long	–	6(1/0) + **	12(1/2) +
CMP	Byte, Word	6(1/0) +	4(1/0) +	–
	Long	6(1/0) +	6(1/0) +	–
DIVS	–	–	158(1/0) + *	–
DIVU	–	–	140(1/0) + *	–
EOR	Byte, Word	–	4(1/0)***	8(1/1) +
	Long	–	8(1/0)***	12(1/2) +
MULS	–	–	70(1/0) + *	–
MULU	–	–	70(1/0) + *	–
OR	Byte, Word	–	4(1/0) +	8(1/1) +
	Long	–	6(1/0) + **	12(1/1) +
SUB	Byte, Word	8(1/0) +	4(1/0) +	8(1/1) +
	Long	6(1/0) + **	6(1/0) + **	12(1/2) +

+ add effective address calculation time
** total of 8 clock periods for instruction if the effective address is register direct
* indicates maximum value
*** only available effective address mode is data register direct

Table 27 IMMEDIATE INSTRUCTION CLOCK PERIODS

Instruction	Size	op #, Dn	op #, An	op #, M
ADDI	Byte, Word	8(2/0)	–	12(2/1) +
	Long	16(3/0)	–	20(3/2) +
ADDQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1) +
	Long	8(1/0)	8(1/0)	12(1/2) +
ANDI	Byte, Word	8(2/0)	–	12(2/1) +
	Long	16(3/0)	–	20(3/1) +
CMPI	Byte, Word	8(2/0)	8(2/0)	8(2/0) +
	Long	14(3/0)	14(3/0)	12(3/0) +
EORI	Byte, Word	8(2/0)	–	12(2/1) +
	Long	16(3/0)	–	20(3/2) +
MOVEQ	Long	4(1/0)	–	–
ORI	Byte, Word	8(2/0)	–	12(2/1) +
	Long	16(3/0)	–	20(3/2) +
SUBI	Byte, Word	8(2/0)	–	12(2/1) +
	Long	16(3/0)	–	20(3/2) +
SUBQ	Byte, Word	4(1/0)	8(1/0)*	8(1/1) +
	Long	8(1/0)	8(1/0)	12(1/2) +

+ add effective address calculation time
*word only

Preliminary

Table 28 SINGLE OPERAND INSTRUCTION CLOCK PERIODS

Instruction	Size	Register	Memory
CLR	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NBCD	Byte	6(1/0)	8(1/1) +
NEG	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NEGX	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
NOT	Byte, Word	4(1/0)	8(1/1) +
	Long	6(1/0)	12(1/2) +
Scc	Byte, False	4(1/0)	8(1/1) +
	Byte, True	6(1/0)	8(1/1) +
TAS	Byte	4(1/0)	10(1/1) +
TST	Byte, Word	4(1/0)	4(1/0)
	Long	4(1/0)	4(1/0) +

+ add effective address calculation time

Table 29 SHIFT AND ROTATE INSTRUCTION CLOCK PERIODS

Instruction	Size	Register	Memory
ASR, ASL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	—
LSR, LSL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	—
ROR, ROL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	—
ROXR, ROXL	Byte, Word	6 + 2n(1/0)	8(1/1) +
	Long	8 + 2n(1/0)	—

Table 30 BIT MANIPULATION INSTRUCTION CLOCK PERIODS

Instruction	Size	Dynamic		Static	
		Register	Memory	Register	Memory
BCHG	Byte	—	8(1/1) +	—	12(2/1) +
	Long	8(1/0)*	—	12(2/0)*	—
BCLR	Byte	—	8(1/1) +	—	12(2/1) +
	Long	10(1/0)*	—	14(2/0)*	—
BSET	Byte	—	8(1/1) +	—	12(2/1) +
	Long	8(1/0)*	—	12(2/0)*	—
BTST	Byte	—	4(1/0) +	—	8(2/0) +
	Long	6(1/0)	—	10(2/0)	—

+ add effective address calculation time

* indicates maximum value

Preliminary

Multi-Precision Instruction Clock Periods

Table 33 indicates the number of clock periods required for the multi-precision instructions. The number of clock periods include the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of bus read and write cycles is shown in parenthesis as (r/w).

In table 33, the headings have the following meaning: Dn = data register operand and M = memory operand.

Miscellaneous Instructions Clock Periods

Table 34 indicates the number of clock periods required for miscellaneous instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Exception Processing Clock Periods

Table 35 indicates the number of clock periods required for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first instruction of the handler routine. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 31 CONDITIONAL INSTRUCTION CLOCK PERIODS

Instruction	Displacement	Trap or Branch Taken	Trap or Branch Not Taken
Bcc	Byte	10(2/0)	8(1/0)
	Word	10(2/0)	12(2/0)
BRA	Byte	10(2/0)	–
	Word	10(2/0)	–
BSR	Byte	18(2/2)	–
	Word	18(2/2)	–
DBcc	cc true	–	12(2/0)
	cc false	10(2/0)	14(3/0)
CHK	–	40(5/3) + *	8(1/0) +
TRAP	–	34(4/3)	–
TRAPV	–	34(5/3)	4(1/0)

\+ add effective address calculation time

* indicates maximum value

Table 32 JMP, JSR, LEA, PEA, MOVEM INSTRUCTION CLOCK PERIODS

Instr	Size	An@	An@+	An@–	An@(d)	An@(d, ix) *	xxx.W	xxx.L	PC@(d)	PC@(d, ix)*
JMP	–	8(2/0)	–	–	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	–	16(2/2)	–	–	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	–	4(1/0)	–	–	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	–	12(1/2)	–	–	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM M ← R	Word	12 + 4n (3 + n/0)	12 + 4n (3 + n/0)	– –	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)	16 + 4n (4 + n/0)	20 + 4n (5 + n/0)	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)
	Long	12 + 8n (3 + 2n/0)	12 + 8n (3 + 2n/0)	– –	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)	16 + 8n (4 + 2n/0)	20 + 8n (5 + 2n/0)	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)
MOVEM R ← M	Word	8 + 5n (2/n)	– –	8 + 5n (2/n)	12 + 5n (3/n)	14 + 5n (3/n)	12 + 5n (3/n)	16 + 5n (4/n)	– –	– –
	Long	8 + 10n (2/2n)	– –	8 + 10n (2/2n)	12 + 10n (3/2n)	14 + 10n (3/2n)	12 + 10n (3/2n)	16 + 10n (4/2n)	– –	– –

n is the number of registers to move

* the size of the index register (ix) does not affect the instruction's execution time

Table 33 MULTI-PRECISION INSTRUCTION CLOCK PERIODS

Instruction	Size	op Dn, Dn	op M, M
ADDX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
CMPM	Byte, Word	–	12(3/0)
	Long	–	20(5/0)
SUBX	Byte, Word	4(1/0)	18(3/1)
	Long	8(1/0)	30(5/2)
ABCD	Byte	6(1/0)	18(3/1)
SBCD	Byte	6(1/0)	18(3/1)

Preliminary

Table 34 MISCELLANEOUS INSTRUCTIONS CLOCK PERIODS

Instruction	Size	Register	Memory	Register ← Memory	Memory ← Register
MOVE from SR	–	6(1/0)	8(1/1) +	–	–
MOVE to CCR	–	12(2/0)	12(2/0) +	–	–
MOVE to SR	–	12(2/0)	12(2/0) +	–	–
MOVEP	Word	–	–	16(2/2)	16(4/0)
	Long	–	–	24(2/4)	24(6/0)
EXG	–	6(1/0)	–	–	–
EXT	Word	4(1/0)	–	–	–
	Long	4(1/0)	–	–	–
LINK	–	16(2/2)	–	–	–
MOVE from USP	–	4(1/0)	–	–	–
MOVE to USP	–	4(1/0)	–	–	–
NOP	–	4(1/0)	–	–	–
RESET	–	132(1/0)	–	–	–
RTE	–	20(5/0)	–	–	–
RTR	–	20(5/0)	–	–	–
RTS	–	16(4/0)	–	–	–
STOP	–	4(0/0)	–	–	–
SWAP	–	4(1/0)	–	–	–
UNLK	–	12(3/0)	–	–	–

+ add effective address calculation time

Table 35 EXCEPTION PROCESSING CLOCK PERIODS

Exception	Periods
Address Error	50(4/7)
Bus Error	50(4/7)
Interrupt	44(5/3)*
Illegal Instruction	34(4/3)
Privileged Instruction	34(4/3)
Trace	34(4/3)

* The interrupt acknowledge bus cycle is assumed to take four external clock periods

Preliminary

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.3 to +7.0	V
Input voltage[3]	−0.3 to +7.0	V
Operating temperature range[2] ,	0 to +70	°C
Storage temperature	−55 to +150	°C

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5.0V \pm 5\%$, $V_{SS} = 0V$; $T_A = 0°C$ to $+70°C$ (see figures 37-39)[4,5]

	PARAMETER	TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IH}	Input high voltage		2.0	V_{CC}	V
V_{IL}	Input low voltage		V_{SS}-0.75	0.8	V
I_{in}	Input leakage current	5.25V			
	BERR, BGACK, BR, DTACK, CLK, IPO0-IPL2, VPA			2.5	μA
	HALT, RESET			20	μA
I_{TSI}	Three-state (off state) input current	2.4V/0.4V			
	AS, A1-A23, D0-D15, FC0-FC2, LDS, R/W, UDS, VMA			20	μA
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$			
	E[6]		V_{CC}-0.75		V
	AS, A1-A23, BG, D0-D15, FC0-FC2, LDS, R/W, UDS, VMA		2.4		V
V_{OL}	Output low voltage				
	HALT	$I_{OL} = 1.6mA$		0.5	V
	A1-A23, BG, FC0-FC2	$I_{OL} = 3.2mA$		0.5	V
	RESET	$I_{OL} = 35.0mA$		0.5	V
	E, AS, D0-D15, LDS, R/W, UDS, VMA	$I_{OL} = 5.3mA$		0.5	V
P_D	Power dissipation	Clock frequency = *MHz		1.5	W
C_{in}	Capacitance	$V_{in} = 0V$, $T_A = 25°C$, frequency = 1MHz		10.0	pF

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. With external pullup resistor of 470 ohms.
7. For a loading capacitance of less than or equal to 500pF, subtract 5ns from the values given in these columns.
8. Actual value depends on clock period.
9. If #47 is satisfied for both DTACK and BERR, #48 can be 0ns.
10. After V_{CC} has been applied for 100ms.
11. If the asynchronous setup time (#47) requirements are satisfied, the DTACK low-to-data setup time (#31) requirements can be ignored. The data must only satisfy the data-in to clock-low setup time (#27) for the following cycle.

Preliminary

AC ELECTRICAL SPECIFICATIONS $V_A = 5VDC$, $T_A = 0°C$ to $70°C$ (see figures 41-45)[4,5]

NUMBER	CHARACTERISTIC	SYMBOL	TENTATIVE LIMITS								UNIT
			4MHz		6MHz		8MHz		10MHz		
			Min	Max	Min	Max	Min	Max	Min	Max	
1	Clock period	t_{cyc}	250	500	167	500	125	500	100	500	ns
2	Clock width low	t_{CL}	115	250	75	250	55	250	45	250	ns
3	Clock width high	t_{CH}	115	250	75	250	55	250	45	250	ns
4	Clock fall time	t_{Cf}		10		10		10		10	ns
5	Clock rise time	t_{Cr}		10		10		10		10	ns
6	Clock low to address	t_{CLAV}		90		80		70		55	ns
6A	Clock high to FC valid	t_{CHFCV}		90		80		70		60	ns
7	Clock high to address data high impedance (maximum)	t_{CHAZx}		120		100		80		70	ns
8	Clock high to address/FC invalid (minimum)	t_{CHAZn}	0		0		0		0		ns
9[7]	Clock high to AS, DS low (maximum)	t_{CHSLx}		80		70		60		55	ns
10	Clock high to AS, DS low (minimum)	t_{CHSLn}	0		0		0		0		ns
11[8]	Address to AS, DS (read) low/AS write	t_{AVSL}	55		35		30		20		ns
11A[8]	FC valid to AS, DS (read) low/AS write	t_{FCVSL}	80		70		60		50		ns
12[7]	Clock low to AS, DS high	t_{CLSH}		90		80		70		55	ns
13[8]	AS, DS high to address/FC invalid	t_{SHAZ}	60		40		30		20		ns
14[8]	AS, DS width low (read)/AS write	t_{SL}	535		337		240		195		ns
14A[8]	DS width low (write)		285		170		115		95		ns
15[7]	AS, DS width high	t_{SH}	285		180		150		105		ns
16	Clock high to AS, DS high impedance	t_{CHSZ}		120		100		80		70	ns
17[8]	AS, DS high to R/W high	t_{SHRH}	60		50		40		20		ns
18[7]	Clock high to R/W high (maximum)	t_{CHRHx}		90		80		70		60	ns
19	Clock high to R/W high (minimum)	t_{CHRHn}	0		0		0		0		ns
20[7]	Clock high to R/W low	t_{CHRL}		90		80		70		60	ns
21[8]	Address valid to R/W low	t_{AVRL}	45		25		20		0		ns
21A[8]	FC valid to R/W low	t_{FCVRL}	80		70		60		50		ns
22[8]	R/W low to DS low (write)	t_{RLSL}	200		140		80		50		ns
23	Clock low to data out valid	t_{CLDO}		90		80		70		55	ns
24	Clock high to R/W, VMA high impedance	t_{CHRZ}		120		100		80		70	ns
25[8]	DS high to data out invalid	t_{SHDO}	60		40		30		20		ns
26[8]	Data out valid to DS low (write)	t_{DOSL}	55		35		30		20		ns
27[11]	Data in to clock low (setup time)	t_{DICL}	30		25		15		15		ns
28[6]	AS, DS high to DTACK high	t_{SHDAH}	0	240	0	160	0	120	0	90	ns
29	DS high to data invalid (hold time)	t_{SHDI}	0		0		0		0		ns
30	AS, DS high to BERR high	t_{SHBEH}	0		0		0		0		ns
31[8]	DTACK low to data in (setup time)	t_{DALDI}		180		120		90		65	ns
32	HALT and RESET input transition time	t_{RHrf}	0	200	0	200	0	200	0	200	ns
33	Clock high to BG low	t_{CHGL}		90		80		70		60	ns
34	Clock high to BG high	t_{CHGH}		90		80		70		60	ns
35	BR low to BG low	t_{BRLGL}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
35	BR low to BG low (figure 43)	t_{BRLGL}	1.5	3.5	1.5	3.5	1.5	3.5	1.5	3.5	Clk Per

Preliminary

AC ELECTRICAL SPECIFICATIONS (Continued) V_A = 5VDC, T_A = 0°C to 70°C (see figures 41-45)[4,5]

NUMBER	CHARACTERISTIC	SYMBOL	TENTATIVE LIMITS								UNIT
			4MHz		6MHz		8MHz		10MHz		
			Min	Max	Min	Max	Min	Max	Min	Max	
36	BR high to BG high	t_{BRHGH}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
37	BGACK low to BG high	t_{GALGH}	1.5	3.0	1.5	3.0	1.5	3.0	1.5	3.0	Clk Per
38	BG low to bus high impedance (AS high)	t_{GLZ}		120		100		80		70	ns
39	BG width high	t_{GH}	1.5		1.5		1.5		1.5		Clk Per
40	Clock low to VMA low	t_{CLVML}		90		80		70		70	ns
41	Clock low to E transition	t_{CLC}		100		85		70		55	ns
42	E output rise and fall time	t_{Erf}		25		25		25		25	ns
43	VMA low to E high	t_{VMLEH}	325		240		200		150		ns
44	AS, DS high to VPA high	t_{SHVPH}	0	240	0	160	0	120	0	90	ns
45	E low to address/VMA/FC invalid	t_{ELAI}	55		35		30		10		ns
46	BGACK width	t_{BGL}	1.5		1.5		1.5		1.5		Clk Per
47[11]	Asynchronous input setup time	t_{ASI}	30		25		20		20		ns
48[9]	BERR low to DTACK low	t_{BELDAL}	50		50		50		50		ns
49	E low to AS, DS invalid	t_{ELSI}	-80		-80		-80		-80		ns
50	E width high	t_{EH}	900		600		450		350		ns
51	E width low	t_{EL}	1400		900		700		550		ns
52	E extended rise time	t_{CIEHX}	80		80		80		80		ns
53	Data hold from clock high	t_{CHDO}	0		0		0		0		ns
54	Data hold from E low (write)	t_{ELDOZ}	60		40		30		20		ns
55	R/W to data bus impedance change	t_{RLDO}	55		35		30		20		ns
56[10]	Halt/RESET pulse width	t_{HRPW}	10		10		10		10		Clk Per

CLOCK TIMING (see figure 40)

CHARACTERISTIC	SYMBOL	4MHz		6MHz		8MHz		10MHz		UNIT
		Min	Max	Min	Max	Min	Max	Min	Max	
Frequency of Operation	F	2.0	4.0	2.0	6.0	2.0	8.0	2.0	10.0	MHz
Cycle Time	t_{cyc}	250	500	167	500	125	500	100	500	ns
Clock Pulse Width	t_{CL} t_{CH}	115 115	250 250	75 75	250 250	55 55	250 250	45 45	250 250	ns
Rise and Fall Times	t_{Cr} t_{Cf}	— —	10 10	— —	10 10	— —	10 10	— —	10 10	ns

POWER CONSIDERATIONS

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \cdot \theta_{JA}) \quad (1)$$

Where:

$T_A \equiv$ Ambient Temperature, °C

$\theta_{JA} \equiv$ Package Thermal Resistance, Junction-to-Ambient, °C/W

$P_D \equiv P_{INT} + P_{I/O}$ (2)

$P_{INT} \equiv I_{CC} \times V_{CC}$, Watts — Chip Internal Power

$P_{I/O} \equiv$ Power Dissipation on Input and Output Pins — User Determined

For most applications $P_{I/O} \ll P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C)$$

Solving equations 1 and 2 for K gives:

$$K = P_D \cdot (T_A + 273°C) + \theta_{JA} \cdot P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

Preliminary

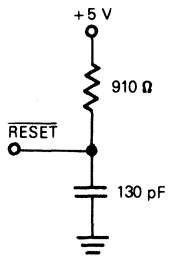


Figure 37. Reset Test Load

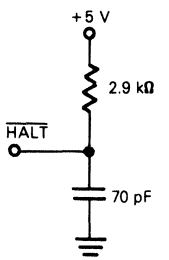
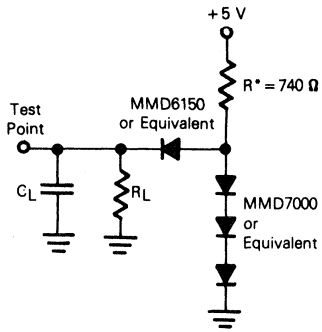


Figure 38. Halt Test Load



C_L = 130 pF
(Includes all Parasitics)
R_L = 6.0 kΩ for
$\overline{AS}$, A1-A23, $\overline{BG}$, D0-D15, E
FC0-FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$
*R = 1.22 kΩ for A1-A23, $\overline{BG}$,
E, FC0-FC2

Figure 39. Test Loads

All timing diagrams should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

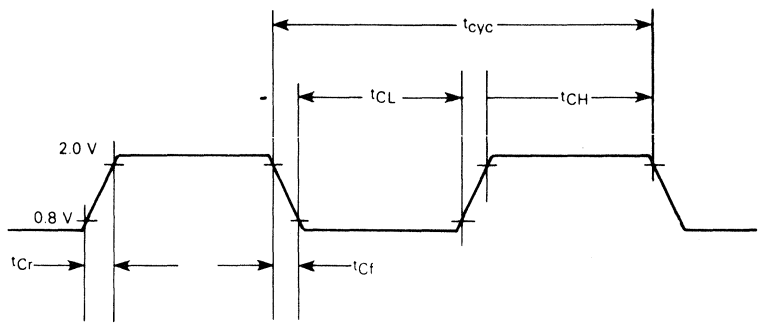
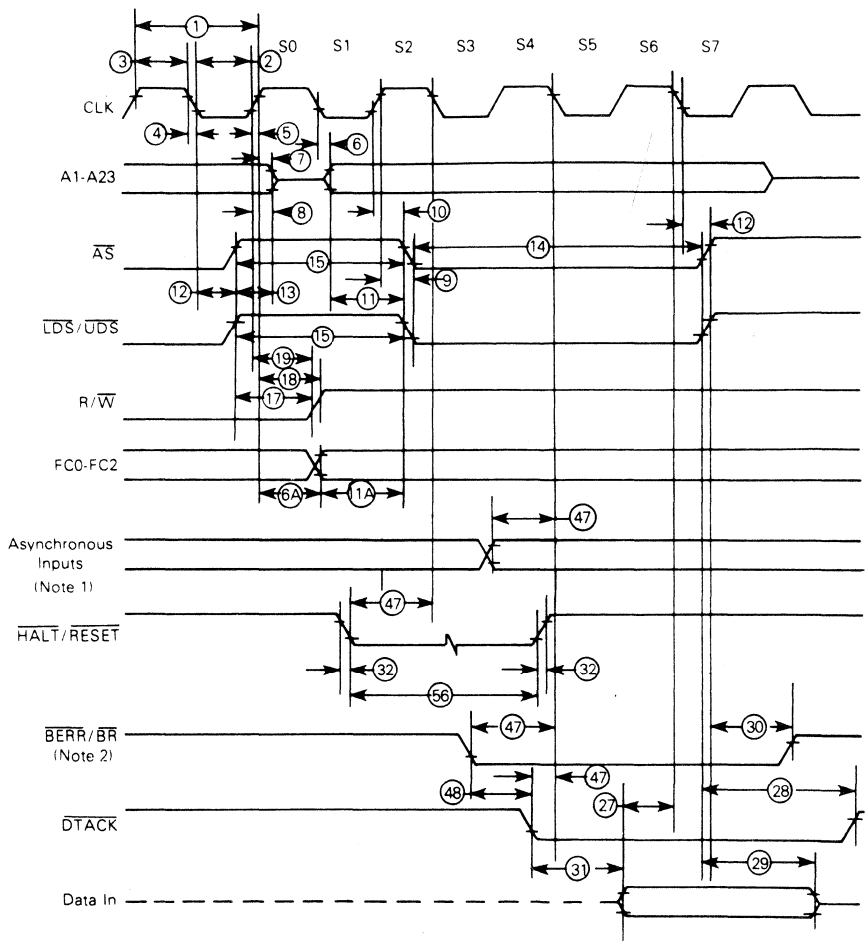


Figure 40. Input Clock Timing

Preliminary



NOTES:

1. Setup time for the asynchronous inputs BGACK, IPL0-IPL2, and VPA guarantees their recognition at the next falling edge of the clock.
2. BR need fall at this time only in order to insure being recognized at the end of this bus cycle.

Figure 41. Read Cycle Timing

Preliminary

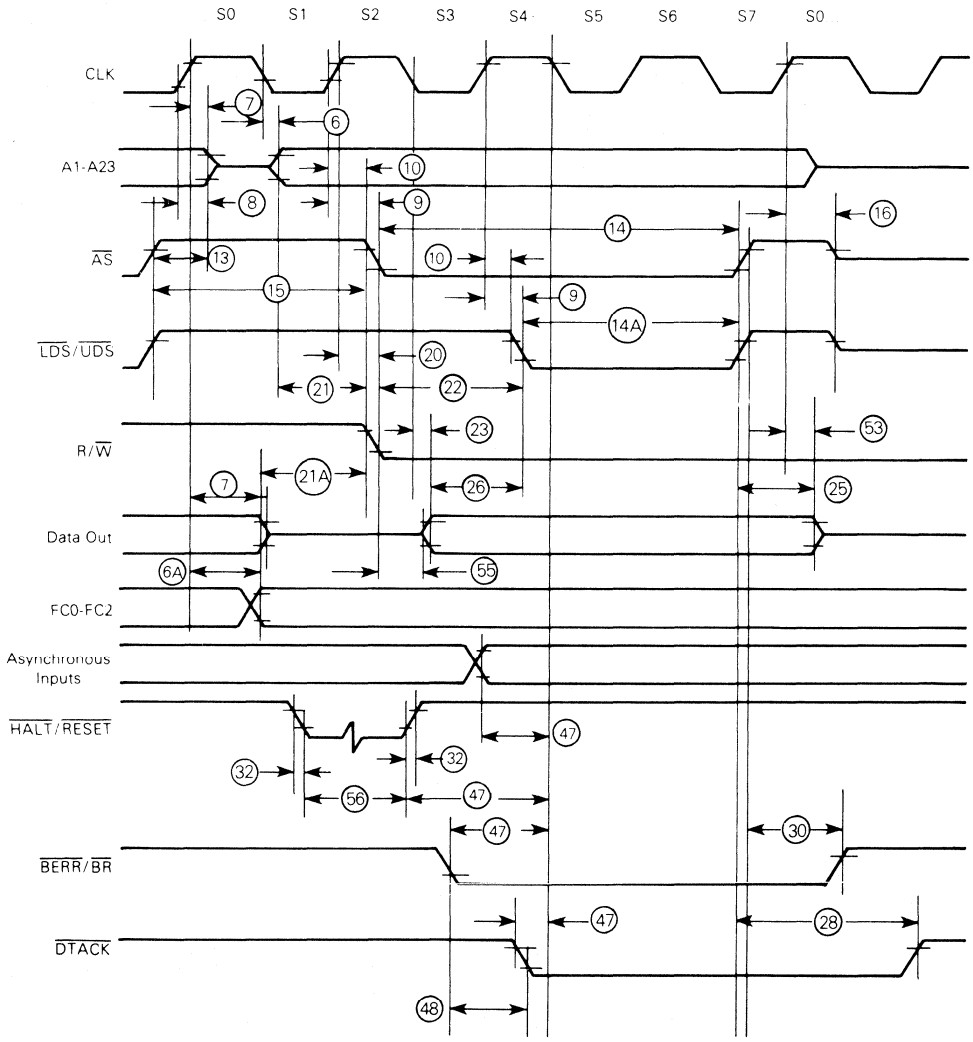
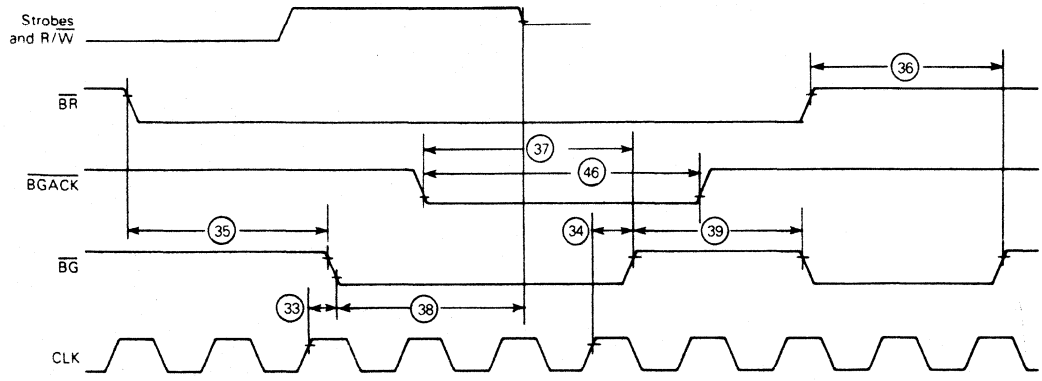


Figure 42. Write Cycle Timing

Preliminary

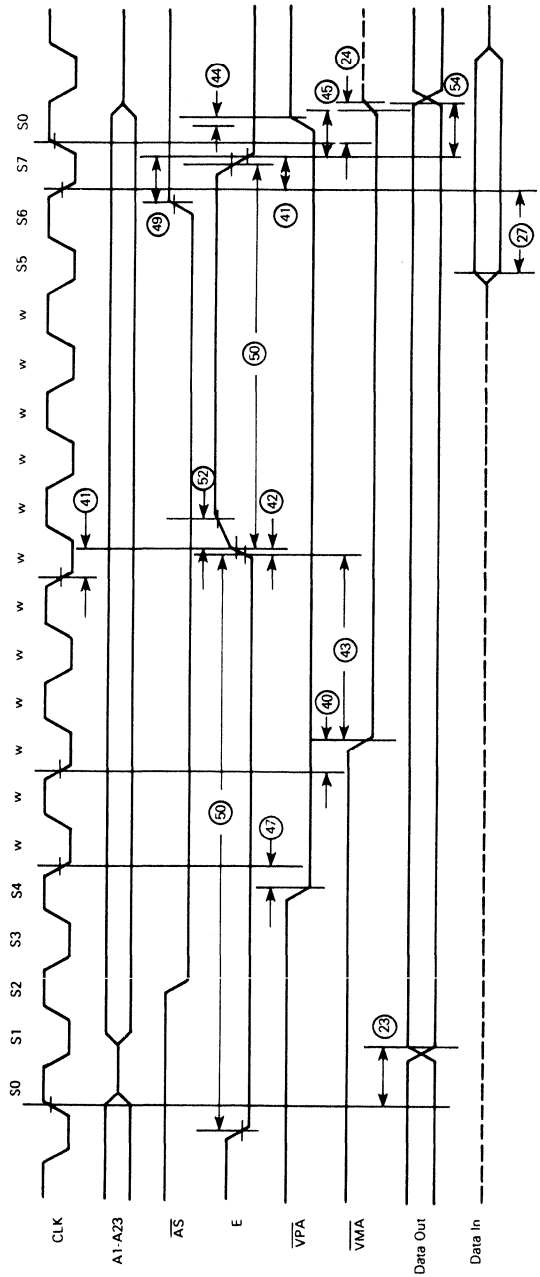


NOTES:

1. Setup time for the asynchronous inputs BERR, BGACK, BR, DTACK, IPL0-IPL2, and VPA guarantees their recognition at the next falling edge of the clock.

Figure 43. AC Electrical Timing—Bus Arbitration

Preliminary



NOTE: This figure represents the best case synchronous timing where $\overline{VPA}$ falls before the third system clock cycle after the falling edge of E.

Figure 44. Synchronous Timing—Best Case

Preliminary

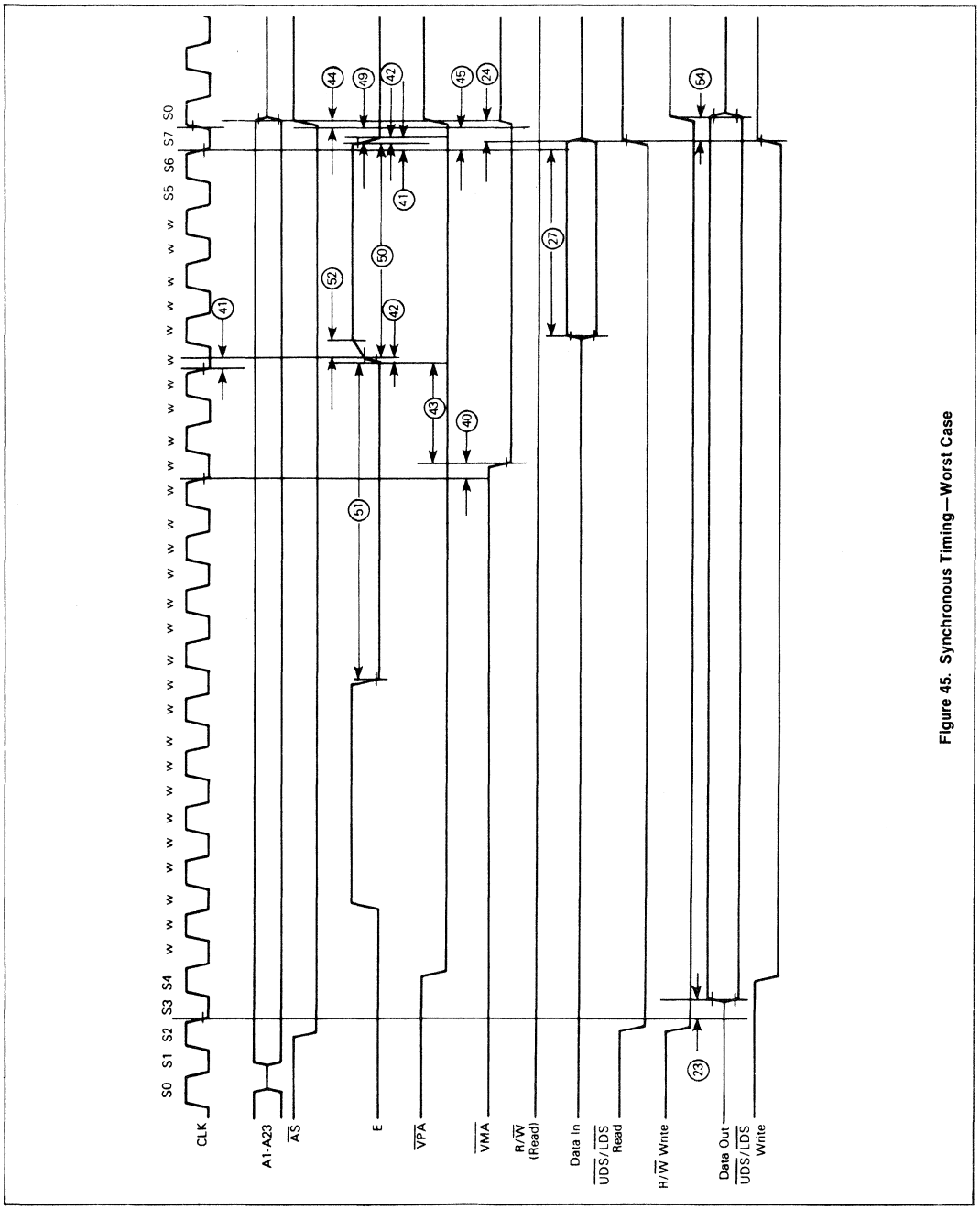


Figure 45. Synchronous Timing—Worst Case

16-Bit Virtual Memory Microprocessor

Originally published by Signetics February 1985

Preliminary

DESCRIPTION

The SCN68010 is the third member of a family of advanced microprocessors from Signetics. Utilizing VLSI technology, the SCN68010 is a fully-implemented 16-bit microprocessor with 32-bit registers, a rich basic instruction set, and versatile addressing modes.

The SCN68010 is fully object code compatible with the earlier members of the S68000 family and has the added features of virtual memory support and enhanced instruction execution timing.

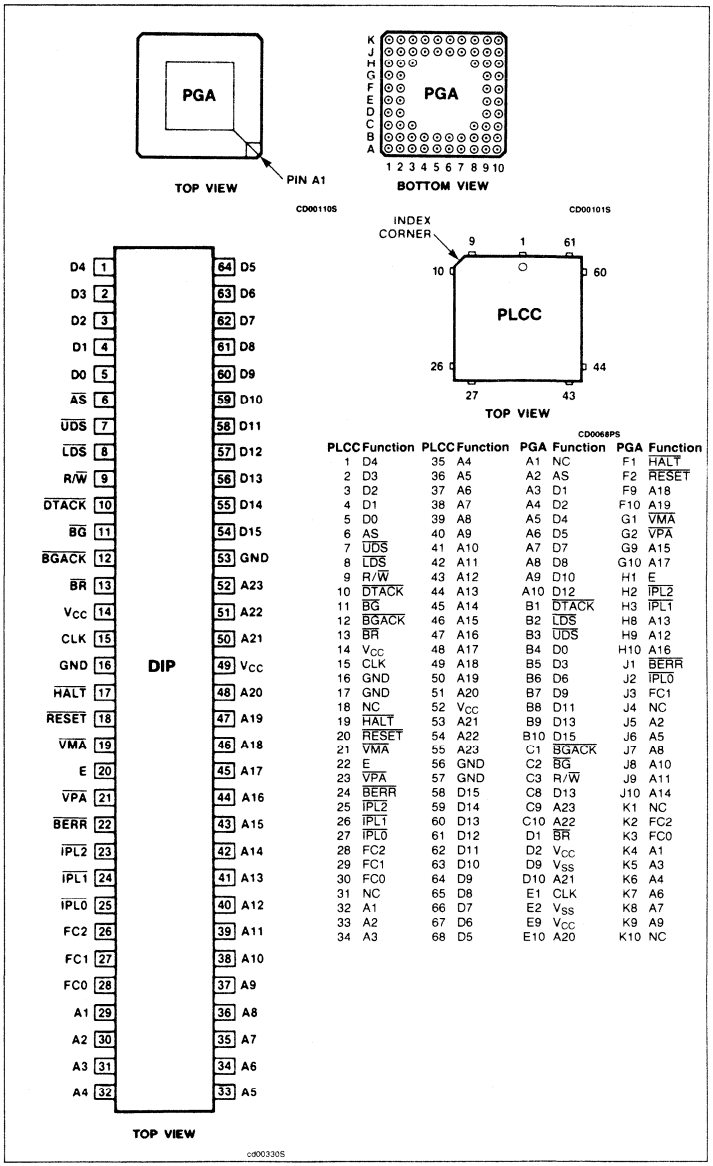
The SCN68010 possesses an asynchronous bus structure with a 24-bit address bus and a 16-bit data bus.

The resources available to the SCN68010 user consist of the following:

- 17 32-bit data and address registers
- 16-megabyte direct addressing range
- Virtual memory machine support
- 57 powerful instruction types
- High performance looping instructions
- Operations on five main data types
- Memory mapped I/O
- 14 addressing modes

As shown in the programming model (figures 1 and 2), the SCN68010 offers 17 32-bit general purpose registers, a 32-bit program counter, a 16-bit status register, a 32-bit vector base register, and two 3-bit alternate function code registers. The first eight registers (D0 – D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) operations. The second set of seven registers (A0 – A6) and the stack pointers (SSP, USP) may be used as software stack pointers and base address registers. In addition, the address registers may be used for word and long word operations. All of the 17 registers may be used as index registers.

PIN CONFIGURATION



CD0068PS

PLCC	Function	PLCC	Function	PGA	Function	PGA	Function
1	D4	35	A4	A1	NC	F1	HALT
2	D3	36	A5	A2	AS	F2	RESET
3	D2	37	A6	A3	D1	F9	A18
4	D1	38	A7	A4	D2	F10	A19
5	D0	39	A8	A5	D4	G1	VMA
6	AS	40	A9	A6	D5	G2	VPA
7	UDS	41	A10	A7	D7	G9	A15
8	LDS	42	A11	A8	D8	G10	A17
9	R/W	43	A12	A9	D10	H1	E
10	DTACK	44	A13	A10	D12	H2	IPL2
11	BG	45	A14	B1	DTACK	H3	IPL1
12	BGACK	46	A15	B2	LDS	H8	A13
13	BR	47	A16	B3	UDS	H9	A12
14	VCC	48	A17	B4	D0	H10	A16
15	CLK	49	A18	B5	D3	J1	BERR
16	GND	50	A19	B6	D6	J2	IPL0
17	GND	51	A20	B7	D9	J3	FC1
18	NC	52	VCC	B8	D11	J4	NC
19	HALT	53	A21	B9	D13	J5	A2
20	RESET	54	A22	B10	D15	J6	A5
21	VMA	55	A23	C1	BGACK	J7	A8
22	E	56	GND	C2	BG	J8	A10
23	VPA	57	GND	C3	R/W	J9	A11
24	BERR	58	D15	C8	D13	J10	A14
25	IPL2	59	D14	C9	A23	K1	NC
26	IPL1	60	D13	C10	A22	K2	FC2
27	IPL0	61	D12	D1	BR	K3	FC0
28	FC2	62	D11	D2	VCC	K4	A1
29	FC1	63	D10	D9	VSS	K5	A3
30	FC0	64	D9	D10	A21	K6	A4
31	NC	65	D8	E1	CLK	K7	A6
32	A1	66	D7	E2	VSS	K8	A7
33	A2	67	D6	E9	VCC	K9	A9
34	A3	68	D5	E10	A20	K10	NC

Preliminary

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C		
	8MHz	10MHz	12.5MHz
Ceramic DIP	SCN68010C8I64	SCN68010CAI64	SCN68010CBI64
Plastic DIP	SCN68010C8N64	SCN68010CAN64	SCN68010CBN64
Plastic LCC	SCN68010C8A68	SCN68010CAA68	SCN68010CBA68
Pin Grid Array	SCN68010C8P68	SCN68010CAP68	SCN68010CBP68

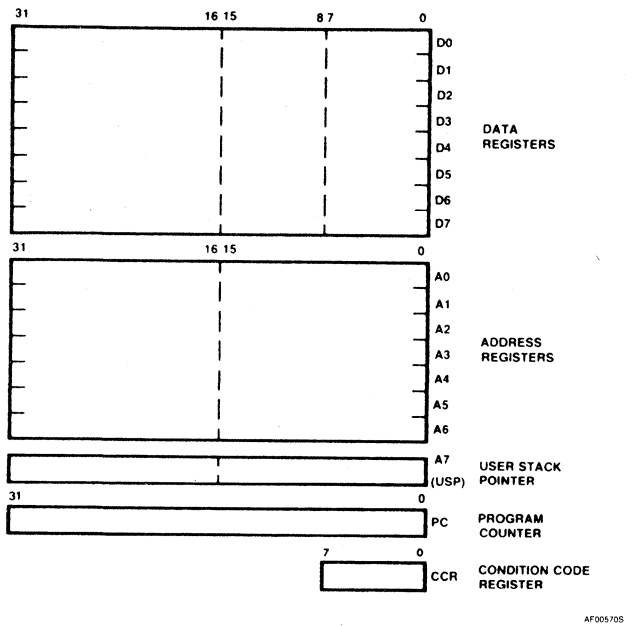


Figure 1. User Programming Model

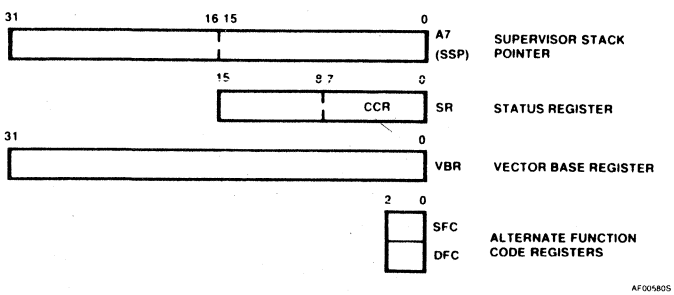


Figure 2. Supervisor Programming Model Supplement

Preliminary

The status register (figure 3) contains the interrupt mask (eight levels available) as well as the condition codes; extend (X), negative (N), zero (Z), overflow (V), and carry (C). Additional status bits indicate that the processor is in the trace (T) mode and in the supervisor (S) or user state.

The vector base register is used to determine the location of the exception vector table in memory to support multiple vector tables. The alternate function code registers allow the supervisor to access user data space or emulate CPU space cycles.

Data Types and Addressing Modes

Five basic data types are supported. These data types are:

- Bits
- BCD digits (4 bits)
- Bytes (8 bits)
- Words (16 bits)
- Long words (32 bits)

In addition, operations on other data types such as memory addresses, status word data, etc., are provided in the instruction set.

The 14 address modes, shown in table 1, include six basic types:

- Register direct
- Register indirect
- Absolute
- Program counter relative
- Immediate
- Implied

Included in the register indirect addressing modes is the capability to do postincrementing, predecrementing, offsetting, and indexing. The program counter relative mode can also be modified via indexing and offsetting.

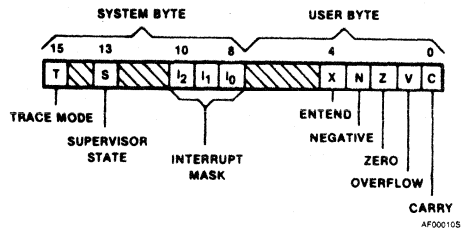


Figure 3. Status Register

Table 1. ADDRESSING MODES

MODE	GENERATION
Register direct addressing	
Data register direct	EA = Dn
Address register direct	EA = An
Absolute data addressing	
Absolute short	EA = (next word)
Absolute long	EA = (next two words)
Program counter relative addressing	
Relative with offset	EA = (PC) + d_{16}
Relative with index and offset	EA = (PC) + (Xn) + d_8
Register indirect addressing	
Register indirect	EA = (An)
Postincrement register indirect	EA = (An), An ← An + N
Predecrement register indirect	An ← An – N, EA = (An)
Register indirect with offset	EA = (An) + d_{16}
Indexed register indirect with offset	EA = (An) + (Xn) + d_8
Immediate data addressing	
Immediate	DATA = next word(s)
Quick immediate	inherent data
Implied addressing	
Implied register	EA = SR, USP, SSP, PC VBR, SFC, DFC

NOTES:
EA = Effective address
An = Address register
Dn = Data register
Xn = Address or data register used as index register
SR = Status register
PC = Program counter
( ) = Contents of
d_8 = 8-bit offset (displacement)
d_{16} = 16-bit offset (displacement)
N = 1 for byte, 2 for word, and 4 for long word. If An is the stack pointer and the operand size is byte, N = 2 to keep the stack pointer on a word boundary.
← = Replaces

Preliminary

Instruction Set Overview

The SCN68010 instruction set is shown in table 2. Some additional instructions are variations, or subsets, of these and they appear in table 3. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. By combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic, and expanded operations (through traps). Also, 33 instructions may be used in the loop mode with certain addressing modes and the DBcc instruction to provide 230 high performance string, block manipulation, and extended arithmetic operations.

Table 2. INSTRUCTION SET

MNEMONIC	DESCRIPTION
ABCD	Add decimal with extend
ADD	Add
AND	Logical AND
ASL	Arithmetic shift left
ASR	Arithmetic shift right
B_{CC}	Branch conditionally
BCHG	Bit test and change
BCLR	Bit test and clear
BRA	Branch always
BSET	Bit test and set
BSR	Branch to subroutine
BTST	Bit test
CHK	Check register against bounds
CLR	Clear operand
CMP	Compare
DB_{CC}	Test condition, decrement and branch
DIVS	Signed divide
DIVU	Unsigned divide
EOR	Exclusive OR
EXG	Exchange registers
EXT	Sign extend
JMP	Jump
JSR	Jump to subroutine
LEA	Load effective address
LINK	Link stack
LSL	Logical shift left
LSR	Logical shift right
MOVE	Move source to destination
MULS	Signed multiply
MULU	Unsigned multiply
NBCD	Negate decimal with extend
NEG	Negate
NOP	No operation
NOT	One's complement
OR	Logical OR
PEA	Push effective address
RESET	Reset external devices
ROL	Rotate left without extend
ROR	Rotate right without extend
ROXL	Rotate left with extend
ROXR	Rotate right with extend
RTD	Return and deallocate
RTE	Return from exception
RTR	Return and restore
RTS	Return from subroutine
SBCD	Subtract decimal with extend
S_{CC}	Set conditional
STOP	Stop
SUB	Subtract
SWAP	Swap data register halves
TAS	Test and set operand
TRAP	Trap
TRAPV	Trap on overflow
TST	Test
UNLK	Unlink

Preliminary

Table 3. VARIATIONS OF INSTRUCTION TYPES

INSTRUCTION TYPE	VARIATION	DESCRIPTION
ADD	ADD	Add
	ADDA	Add address
	ADDQ	Add quick
	ADDI	Add immediate
	ADDX	Add with extend
AND	AND	Logical AND
	ANDI	AND immediate
	ANDI to CCR	AND immediate to condition codes
	ANDI to SR	AND immediate to status register
CMP	CMP	Compare
	CMPA	Compare address
	CMPM	Compare memory
	COMPI	Compare immediate
EOR	EOR	Exclusive OR
	EORI	Exclusive OR immediate
	EORI to CCR	Exclusive OR immediate to condition codes
	EORI to SR	Exclusive OR immediate to status register
MOVE	MOVE	Move source to destination
	MOVEA	Move address
	MOVEC	Move control register
	MOVEM	Move multiple registers
	MOVEP	Move peripheral data
	MOVEQ	Move quick
	MOVES	Move alternate address space
	MOVE from SR	Move from status register
	MOVE to SR	Move to status register
	MOVE from CCR	Move from condition codes
	MOVE to CCR	Move to condition codes
	MOVE USP	Move user stack pointer
NEG	NEG	Negate
	NEGX	Negate with extend
OR	OR	Logical OR
	ORI	OR immediate
	ORI to CCR	OR immediate to condition codes
	ORI to SR	OR immediate to status register
SUB	SUB	Subtract
	SUBA	Subtract address
	SUBI	Subtract immediate
	SUBQ	Subtract quick
	SUBX	Subtract with extend

Virtual Memory/Machine Concepts

In most systems using the SCN68010 as the central processor, only a fraction of the 16 megabyte address space will actually contain physical memory. However, by using virtual memory techniques the system can be made to appear to the user to have 16 megabytes of physical memory available to him/her. These techniques have been used for several years in large mainframe computers and more recently in minicomputers and now, with the SCN68010, can be fully supported in microprocessor-based systems.

In a virtual memory system, a user program can be written as though it has a large amount of memory available to it when only a small amount of memory is physically present in the system. In a similar fashion, a system can be designed in such a manner as to allow user programs to access other types of devices that are not physically present in the system such as tape drives, disk drives, printers, or CRTs. With proper software emulation, a physical system can be made to appear to a user program as any other computer system and the program may be given full access to all of the resources of that emulated system. Such an emulated system is called a virtual machine.

Virtual Memory

The basic mechanisim for supporting virtual memory in computers is to provide only a limited amount of high-speed physical memory that can be accessed directly by the processor while maintaining an image of a much larger "virtual" memory on secondary storage devices such as large capacity disk drives. When the processor attempts to access a location in the virtual memory map that is not currently residing in physical memory (referred to as a page fault), the access to that location is temporarily suspended while the necessary data is fetched from the secondary storage and placed in physical memory; the suspended access is then completed. The SCN68010 provides hardware support for virtual memory with the capability of suspending an instruction's execution when a bus error is signaled and then completing the instruction after the physical memory has been updated as necessary.

The SCN68010 uses instruction continuation rather than instruction restart to support virtual memory. With instruction restart, the processor must remember the exact state of the system before each instruction is started in order to restore that state if a page fault occurs during its execution. Then, after the page fault has been repaired, the entire instruction that caused the fault is reexecuted. With instruction continuation, when a page fault occurs the processor stores its internal state and then, after the page fault is repaired, restores that internal state and continues execution of the instruction. In order for the SCN68010 to utilize instruction continuation, it stores its internal state on the supervisor stack when a bus cycle is terminated with a bus error signal. It then loads the program counter from vector table entry number two (offset $008) and resumes program execution at that new address. When the bus error exception handler routine has completed execution, an RTE instruction is executed which reloads the SCN68010 with the internal state stored on the stack, re-runs the faulted bus cycle, and continues the suspended instruction. Instruction continuation has the additional advantage of allowing hardware support for virtual I/O devices. Since virtual registers may be simulated in the memory map, an access to such a register will cause a fault and the function of the register can be emulated by software.

Virtual Machine

One typical use for a virtual machine system is in the development of software such as an operating system for another machine with

hardware also under development and not available for programming use. In such a system, the governing operating system (OS) emulates the hardware of the new system and allows the new OS to be executed and debugged as though it were running on the new hardware. Since the new OS is controlled by the governing OS, the new one must execute at a lower privilege level than the governing OS so that any attempts by the new OS to use virtual resources that are not physically present, and should be emulated, will be trapped by the governing OS and handled in software. In the SCN68010, a virtual machine may be fully supported by running the new OS in the user mode and the governing OS in the supervisor mode so that any attempts to access supervisor resources or execute privileged instructions by the new OS will cause a trap to the governing OS.

In order to fully support a virtual machine, the SCN68010 must protect the supervisor resources from access by user programs. The one supervisor resource that is not fully protected in the SCN68000 is the system byte of the status register. In the SCN68000, the MOVE from SR instruction allows user programs to test the S bit (in addition to the 1 bit and interrupt mask) and thus determine that they are running in the user mode. For full virtual machine support, a new OS must not be aware of the fact that it is running in the user mode and thus should not be allowed to access the S bit. For this reason, the MOVE from SR instruction on the SCN68010 is a privileged instruction and the MOVE from CCR instruction has been added to allow user programs unhindered access to the condition codes. By making the MOVE from SR instruction privileged, when the new OS attempts to access the S bit, a trap to the governing OS will occur and the SR image passed to the new OS by the governing OS will have the S bit set.

DATA ORGANIZATION AND ADDRESSING CAPABILITIES

This section contains a description of the registers and the data organization of the SCN68010.

Operand Size

Operand sizes are defined as follows: a byte equals 8 bits, a word equals 16 bits, and a long word equals 32 bits. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Implicit instructions support some subset of all three sizes.

Data Organization in Registers

The eight data registers support data operands of 1, 8, 16, or 32 bits. The seven address registers and the stack pointers support address operands of 32 bits. The four control registers support data of 1, 3, 8, 16, or 32 bits depending on the register specified.

Data Registers

Each data register is 32 bits wide. Byte operands occupy the low order 8 bits, word operands the low order 16 bits, and long word operands the entire 32 bits. The least significant bit is addressed as bit zero; the most significant bit is addressed as bit 31.

When a data register is used as either a source or destination operand, only the appropriate low order portion is changed; the remaining high order portion is neither used nor changed.

Address Registers

Each address register and stack pointer is 32 bits wide and holds a full 32-bit address. Address registers do not support the sized operands. Therefore, when an address register is used as a source operand, either the low order word or the entire long word operand is used depending upon the operation size. When an address register is used as the destination operand, the entire register is affected regardless of the operation size. If the operation size is word, any other operands are sign extended to 32 bits before the operation is performed.

Control Registers

The status register (SR) is 16 bits wide with the lower byte being accessed as the condition code register (CCR). Not all 16 bits of the SR are defined and will be read as zeroes and ignored when written. Operations to the CCR are word operations; however, the upper byte will be read as all zeros and ignored when written.

The vector base register (VBR) is 32 bits wide and holds a full 32-bit address. All operations involving the VBR are long word operations regardless of whether it is the source or destination operand.

The alternate function code registers (SFC and DFC) are three bits wide and contain the function code values placed on FC0 – FC2 during the operand read or write of a MOVES instruction. All transfers to or from the alternate function code registers are 32 bits although the upper 29 bits will be read as zeroes and ignored when written.

Data Organization in Memory

The data types supported by the SCN68010 are: bit data, integer data of 8, 16, or 32 bits, 32-bit addresses and binary coded decimal data. Each of these data types is put in memory, as shown in figure 4. The numbers indicate the order in which the data would be accessed from the processor.

Bytes are individually addressable with the high order byte having an even address the same as the word, as shown in figure 5. The low order byte has an odd address that is one count higher than the word address. Instructions and word or long word data are accessed only on word (even byte) boundaries. If a long word datum is located at address n (n even), then the low-order word of that datum is located at address n + 2.

Addressing

Instructions for the SCN68010 contain two kinds of information: the type of function to be performed and the location of the operand(s) on which to perform that function. The methods used to locate (address) the operand(s) are explained in the following paragraphs.

Instructions specify an operand location in one of three ways:

Register Specification – the number of the register is given in the register field of their instruction.

Effective Address – use of the different effective addressing modes.

Implicit Reference – the definition of certain instructions implies the use of specific registers.

Preliminary

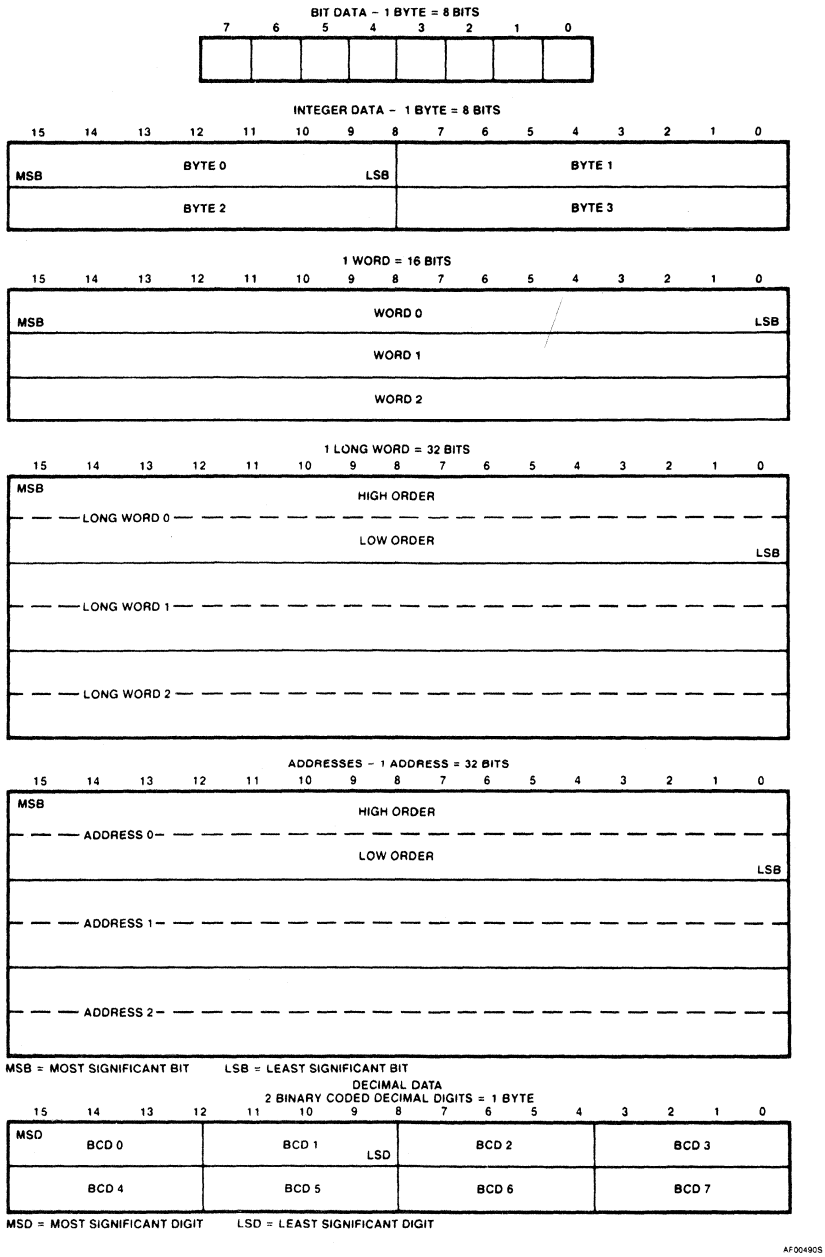


Figure 4. Memory Data Organization

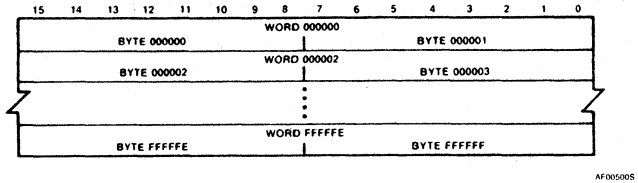


Figure 5. Word Organization in Memory

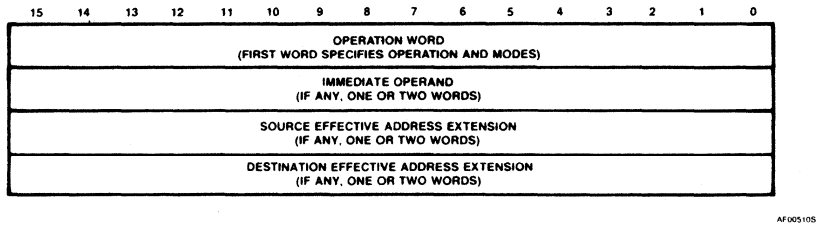


Figure 6. Instruction Operation Word General Format

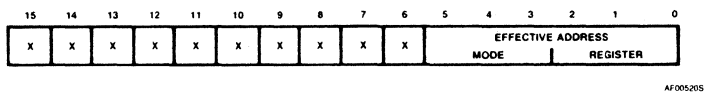


Figure 7. Single-Effective-Address Instruction Operation Word

Instruction Format

Instructions are from one to five words in length as shown in figure 6. The length of the instruction and the operation to be performed is specified by the first word of the instruction which is called the operation word. The remaining words further specify the operands. These words are either immediate operands or extensions to the effective address mode specified in the operation word.

Program/Data References

The SCN68010 separates memory references into two classes: program references and data references. Program references, as the name implies, are references to that section of memory that contains the program being executed. Data references refer to that section of memory that contains data. Generally, operand reads are from the data space. All operand writes are to the data space.

Register Specification

The register field within an instruction specifies the register to be used. Other fields within the instruction specify whether the register selected is an address or data register and how the register is to be used.

Effective Address

Most instructions specify the location of an operand by using the effective address field in the operation word. For example, figure 7 shows the general format of the single-effective-address instruction operation word. The effective address is composed of two 3-bit fields: the mode field and the register field. The value in the mode field selects the different address modes. The register field contains the number of a register.

The effective address field may require additional information to fully specify the operand. This additional information, called the effective address extension, is contained in the following word or words and is considered part of the instruction, as shown in figure 6. The effective address modes are grouped into three categories: register direct, memory addressing, and special.

Register Direct Modes

These effective addressing modes specify that the operand is in one of sixteen general purpose registers or one of four control registers.

Data Register Direct — The operand is in the data register specified by the effective address register field.

Address Register Direct — The operand is in the address register specified by the effective address register field.

Memory Address Modes

These effective addressing modes specify that the operand is in memory and provide the specific address of the operand.

Address Register Indirect — The address of the operand is in the address register specified by the register field. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Address Register Indirect with Post Increment — The address of the operand is in the address register specified by the register field. After the operand address is used, it is incremented by one, two, or four depending upon whether the size of the operand is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is incremented by two rather than one to keep the stack pointer on a

word boundary. The reference is classified as a data reference.

Address Register Indirect with Predecrement — The address of the operand is in the address register specified by the register field. Before the operand address is used, it is decremented by one, two, or four depending upon whether the operand size is byte, word, or long word. If the address register is the stack pointer and the operand size is byte, the address is decremented by two rather than one to keep the stack pointer on a word boundary. The reference is classified as a data reference.

Address Register Indirect with Displacement — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the address register and the sign-extended 16-bit displacement integer in the extension word. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Address Register Indirect with Index — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the address register, the sign-extended displacement integer in the low order eight bits of the extension word, and the contents of the index register. The index may be specified as the sign extended low-order word or the long word in the index register. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Special Address Modes

The special address modes use the effective address register field to specify the special addressing mode instead of a register number.

Absolute Short Address — This addressing mode requires one word of extension. The address of the operand is in the extension word. The 16-bit address is sign extended before it is used. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Absolute Long Address — This addressing mode requires two words of extension. The address of the operand is developed by the concatenation of the extension words. The high order part of the address is the first extension word; the low order part of the address is the second extension word. The reference is classified as a data reference with the exception of the jump and jump-to-subroutine instructions.

Program Counter with Displacement — This addressing mode requires one word of extension. The address of the operand is the sum of the address in the program counter and the sign-extended 16-bit displacement integer in the extension word. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Table 4. EFFECTIVE ADDRESS ENCODING SUMMARY

ADDRESSING MODE	MODE	REGISTER
Data register direct	000	Register number
Address register direct	001	Register number
Address register indirect	010	Register number
Address register indirect with postincrement	011	Register number
Address register indirect with predecrement	100	Register number
Address register indirect with displacement	101	Register number
Address register indirect with index	110	Register number
Absolute short	111	000
Absolute long	111	001
Program counter with displacement	111	010
Program counter with index	111	011
Immediate	111	100

Program Counter with Index — This addressing mode requires one word of extension. The address is the sum of the address in the program counter, the sign-extended displacement integer in the lower eight bits of the extension word, and the contents of the index register. The index may be specified as the sign extended low-order word or the long word in the index register. The value in the program counter is the address of the extension word. The reference is classified as a program reference.

Immediate Data — This addressing mode requires either one or two words of extension depending on the size of the operation.

Byte Operation – operand is in the low order byte of extension word

Word Operation – operand is in the extension word

Long Word – Operation operand is in the two extension words, high order 16 bits are in the first extension word, low order 16 bits are in the second extension word.

Implicit Reference — Some instructions make implicit reference to the program counter (PC), the system stack pointer (SP), the supervisor stack pointer (SSP), the user stack pointer (USP), the status register (SR), the condition code register (CCR), the vector base register (VBR), or the alternate function code registers (SFC or DFC).

A selected set of instructions may reference the status register by means of the effective address field. These are:

ANDI to CCR	ORI to SR
ANDI to SR	MOVE to CCR
EORI to CCR	MOVE to SR
EORI to SR	MOVE from SR
ORI to CCR	

Effective Address Encoding Summary

Table 4 is a summary of the effective addressing modes discussed in the previous paragraphs.

System Stack

The system stack is used implicitly by many instructions; user stacks and queues may be created and maintained through the addressing modes. Address register seven (A7) is the system stack pointer (SP). The system stack pointer is either the supervisor stack pointer (SSP) or the user stack pointer (USP), depending on the state of the S bit in the status register. If the S bit indicates supervisor state, the SSP is the active system stack pointer and the USP cannot be referenced as an address register. If the S bit indicates user state, the USP is the active system stack pointer, and the SSP cannot be referenced. Each system stack fills from high memory to low memory.

INSTRUCTION SET SUMMARY

This section contains an overview of the form and structure of the SCN68010 instruction set. The instructions form a set of tools that include all the machine functions to perform the following operations:

Data movement	Bit manipulation
Integer arithmetic	Binary code decimal
Logical	Program control
Shift and rotate	System control

The complete range of instruction capabilities combined with the flexible addressing modes

described previously provide a very flexible base for program development.

Data Movement Operations

The basic method of data acquisition (transfer and storage) is provided by the move (MOVE) instruction. The move instruction and the effective addressing modes allow both address and data manipulation. Data movement instructions allow byte, word, and long word operands to be transferred from memory to memory, memory to register, register to memory, and register to register. Address movement instructions allow word and long word operand transfers and ensure that only legal address manipulations are executed. In addition to the general move instruction there are several special data movement instructions: move multiple registers (MOVEM), move peripheral data (MOVEP), exchange registers (EXG), load effective address (LEA), push effective address (PEA), link stack (LINK), unlink stack (UNLK), move quick (MOVEQ), move control register (MOVECS), and move alternate address space (MOVES). Table 5 is a summary of the data movement operations.

Integer Arithmetic Operations

The arithmetic operations include the four basic operations of add (ADD), subtract (SUB), multiply (MUL), and divide (DIV) as well as arithmetic compare (CMP), clear (CLR), and negate (NEG). The add and subtract instructions are available for both address and data operations, with data operations accepting all operand sizes. Address operations are limited to legal address size operands (16 or 32 bits). Data, address, and memory compare operations are also available. The clear and negate instructions may be used on all sizes of data operands.

The multiply and divide operations are available for signed and unsigned operands using word multiply to produce a long word product, and a long word dividend with word divisor to produce a word quotient with a word remainder.

Multiprecision and mixed size arithmetic can be accomplished using a set of extended instructions. These instructions are: add extended (ADDX), subtract extended (SUBX), sign extend (EXT), and negate binary with extend (NEGX).

Table 5. DATA MOVEMENT OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
EXG	32	Rx ↔ Ry
LEA	32	EA → An
LINK	–	(An) → –(SP) (SP) → An (SP) + displacement → SP
MOVE	8, 16, 32	(EA)s → EAd
MOVEC	32	(Rn) → Cr (Cr) → Rn
MOVEM	16, 32	(EA) → An, Dn An, Dn → EA
MOVES	8, 16, 32	(EA) → Rn (Rn) → EA
MOVEP	16, 32	d(An) → Dn Dn → d(An)
MOVEQ	8	#xxx → Dn
PEA	32	EA → –(SP)
SWAP	32	Dn[31:16] ↔ Dn[15:0]
UNLK	–	An → Sp (SP) + → An

NOTES:
s = source
d = destination
[ ] = bit numbers
–( ) = indirect with predecrement
( ) + = indirect with postdecrement
= immediate data

Table 6. INTEGER ARITHMETIC OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ADD	8, 16, 32 16, 32	(Dn) + (EA) → Dn (EA) + (Dn) → EA (EA) + #xxx → EA (An) + (EA) → An
ADDX	8, 16, 32 16, 32	(Dx) + (Dy) + X → Dx –(Ax) + –(Ay) + X → (Ax)
CLR	8, 16, 32	0 → (EA)
CMP	8, 16, 32 16, 32	(Dn) – (EA) (EA) – #xxx (Ax) + –(Ay) + (An) – (EA)
DIVS	32÷16	(Dn)/(EA) → Dn
DIVU	32÷16	(Dn)/(EA) → Dn
EXT	8 → 16 16 → 32	$(Dn)_8 \rightarrow Dn_{16}$ $(Dn)_{16} \rightarrow Dn_{32}$
MULS	16 x 16 → 32	(Dn) x (EA) → Dn
MULU	16 x 16 → 32	(Dn) x (EA) → Dn
NEG	8, 16, 32	0 – (EA) → EA
NEGX	8, 16, 32	0 – (EA) – X → EA
SUB	8, 16, 32 16, 32	(Dn) – (EA) → Dn (EA) – Dn → EA (EA) – #xxx → EA (An) – (EA) → An
SUBX	8, 16, 32	(Dx) – (Dy) – X → Dx –(Ax) – –(Ay) – X → (Ax)
TAS	8	[EA] – 0, 1 → EA[7]
TST	8, 16, 32	(EA) – 0

NOTES:
[ ] = bit number
= immediate data
–( ) = indirect with predecrement
( ) + = indirect with postdecrement

Preliminary

A test operand (TST) instruction that will set the condition codes as a result of a compare of the operand with zero is also available. Test and set (TAS) is a synchronization instruction useful in multiprocessor systems. Table 6 is a summary of the integer arithmetic operations.

Logical Operations

Logical operation instructions AND, OR, EOR, and NOT are available for all sizes of integer data operands. A similar set of immediate instructions (ANDI, ORI, and EORI) provide these logical operations with all sizes of immediate data. Table 7 is a summary of the logical operations.

Table 7. LOGICAL OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
AND	8, 16, 32	(Dn) ∧ (EA) → Dn (EA) ∧ (Dn) → EA (EA) ∧ #xxx → EA
OR	8, 16, 32	(Dn) ∨ (EA) → Dn (EA) ∨ (Dn) → EA (EA) ∨ #xxx → EA
EOR	8, 16, 32	(EA) ⊕ (Dy) → EA (EA) ⊕ #xxx → EA
NOT	8, 16, 32	~(EA) → EA

NOTES:
= immediate data
~ = invert
∧ = logical AND
∨ = logical OR
⊕ = logical exclusive OR

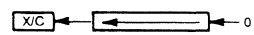
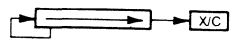
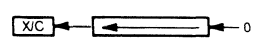
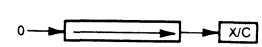
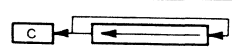
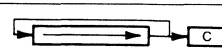
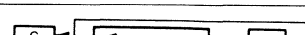
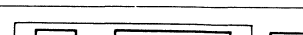
Shift and Rotate Operations

Shift operations in both directions are provided by the arithmetic shift instructions ASR and ASL and logical shift instructions LSR and LSL. The rotate instructions (with and without extend) available are ROXR, ROXL, ROR, and ROL. All shift and rotate operations can be performed in either registers or memory. Register shifts and rotates support all operand sizes and allow a shift count specified in a data register.

Memory shifts and rotates are for word operands only and allow only single-bit shifts or rotates.

Table 8 is a summary of the shift and rotate operations.

Table 8. SHIFT AND ROTATE OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ASL	8, 16, 32	
ASR	8, 16, 32	
LSL	8, 16, 32	
LSR	8, 16, 32	
ROL	8, 16, 32	
ROR	8, 16, 32	
ROXL	8, 16, 32	
ROXR	8, 16, 32	

Preliminary

Bit Manipulation Operations

Bit manipulation operations are accomplished using the following instructions: bit test (BTST), bit test and set (BSET), bit test and clear (BCLR), and bit test and change (BCHG). Table 9 is a summary of the bit manipulation operations. (Z is bit 2 of the status register.)

Table 9. BIT MANIPULATION OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
BTST	8, 32	~bit of (EA) → Z
BSET	8, 32	~bit of (EA) → Z 1 → bit of EA
BCLR	8, 32	~bit of (EA) → Z 0 → bit of EA
BCHG	8, 32	~bit of (EA) → Z ~bit of (EA) → bit of EA

NOTE:
~ = invert

Binary Coded Decimal Operations

Multiprecision arithmetic operations on binary coded decimal numbers are accomplished using the following instructions: add decimal with extend (ABCD), subtract decimal with extend (SBCD), and negate decimal with extend (NBCD). Table 10 is a summary of the binary coded decimal-operations.

Table 10. BINARY CODED DECIMAL OPERATIONS

INSTRUCTION	OPERAND SIZE	OPERATION
ABCD	8	$(Dx)_{10} + (Dy)_{10} + X \rightarrow Dx$ $-(Ax)_{10} + -(Ay)_{10} + X \rightarrow (Ax)$
SBCD	8	$(Dx)_{10} - (Dy)_{10} - X \rightarrow Dx$ $-(Ax)_{10} - -(Ay)_{10} - X \rightarrow (Ax)$
NBCD	8	$0 - (EA)_{10} - X \rightarrow (EA)$

NOTES:
– = indirect with predecrement.
+ = indirect with postdecrement

Program Control Operations

Program control operations are accomplished using a series of conditional and unconditional branch instructions and return instructions. These instructions are summarized in table 11.

The conditional instructions provide setting and branching for the following conditions:

CC — carry clear
CS — carry set
EQ — equal
F — never true
GE — greater or equal
GT — greater than
HI — high
LE — less or equal
LS — low or same
LT — less than
MI — minus
NE — not equal
PL — plus
T — always true
VC — no overflow
VS — overflow

Table 11. PROGRAM CONTROL OPERATIONS

INSTRUCTION	OPERATION
Conditional	
B_{CC}	Branch conditionally (14 conditions) 8- and 16-bit displacement
DB_{CC}	Test condition, decrement, and branch 16-bit displacement
S_{CC}	Set byte conditionally (16 conditions)
Unconditional	
BRA	Branch always 8- and 16-bit displacement
BSR	Branch to subroutine 8- and 16-bit displacement
JMP	Jump
JSR	Jump to subroutine
Returns	
RTD	Return from subroutine and deallocate stack
RTR	Return and restore condition codes
RTS	Return from subroutine

Preliminary

System Control Operations

System control operations are accomplished by using privileged instructions, trap generating instructions, and instructions that use or modify the condition code register. These instructions are summarized in table 12.

Table 12. SYSTEM CONTROL OPERATIONS

INSTRUCTION	OPERATION
Privileged	
ANDI to SR	Logical AND to status register
EORI to SR	Logical EOR to status register
MOVE EA to SR	Load new status register
MOVE SR to EA	Store status register
MOVE USP	Move user stack pointer
MOVEC	Move control register
MOVES	Move alternate address space
ORI to SR	Logical OR to status register
RESET	Reset external devices
RTE	Return from exception
STOP	Stop program execution
Trap Generating	
CHK	Check data register against upper bounds
TRAP	Trap
TRAPV	Trap on overflow
Condition Code Register	
ANDI to CCR	Logical AND to condition codes
EORI to CCR	Logical EOR to condition codes
MOVE EA to CCR	Load new condition codes
MOVE CCR to EA	Store condition codes
ORI to CCR	Logical OR to condition codes

SIGNAL AND BUS OPERATION DESCRIPTION

This section contains a brief description of the input and output signals. A discussion of bus operation during the various machine cycles and operations is also given.

NOTE

The terms **assertion** and **negation** will be used extensively. This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true, independent of whether that level is represented by a high or low voltage. The term negate or negation is used to indicate that a signal is inactive or false.

Signal Description

The input and output signals can be functionally organized into the groups shown in figure 8. The following paragraphs provide a brief description of the signals and a reference (if applicable) to other paragraphs that contain more detail about the function being performed.

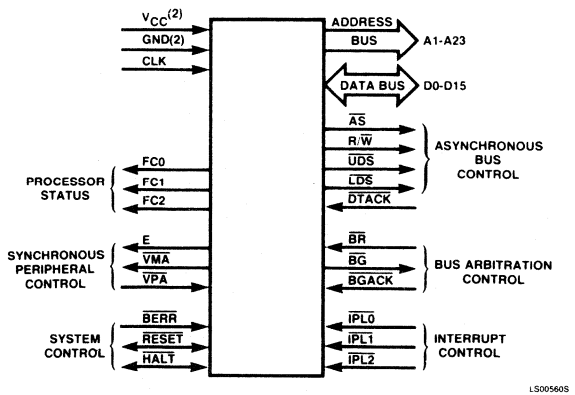


Figure 8. Input and Output Signals

Preliminary

Address Bus (A1 through A23)
This 23-bit, unidirectional, three-state bus is capable of addressing 8 megawords of data. It provides the address for bus operation during all cycles except CPU space cycles.

Data Bus (D0 through D15)
This 16-bit, bidirectional, three-state bus is the general purpose data path. It can transmit and accept data in either word or byte length.

Asynchronous Bus Control
Asynchronous data transfers are handled using the following control signals: address strobe, read/write, upper and lower data strobes, and data transfer acknowledge. These signals are explained in the following paragraphs.

Address Strobe ($\overline{AS}$) — This signal indicates that there is a valid address on the address bus.

Read/Write (R/$\overline{W}$) — This signal defines the data bus transfer as a read or write cycle. The R/$\overline{W}$ signal also works in conjunction with the data strobes as explained in the following paragraph.

Upper and Lower Data Strobe (UDS, LDS) — These signals control the flow of data on the data bus, as shown in table 13. When the R/$\overline{W}$ line is high, the processor will read from the data bus as indicated. When the R/$\overline{W}$ line is low, the processor will write to the data bus as shown.

Data Transfer Acknowledge ($\overline{DTACK}$) — This input indicates that the data transfer is completed. When the processor recognizes $\overline{DTACK}$ during a read cycle, data is latched one clock cycle later and the bus cycle terminated. When $\overline{DTACK}$ is recognized during a write cycle, the bus cycle is terminated. Refer to **Asynchronous Versus Synchronous Operation.**

Bus Arbitration Control
The three signals, bus request, bus grant, and bus grant acknowledge, form a bus arbitration circuit to determine which device will be the bus master device.

Bus Request ($\overline{BR}$) — This input is wire ORed with all other devices that could be bus masters. This input indicates to the processor that some other device desires to become the bus master.

Bus Grant ($\overline{BG}$) — This output indicates to all other potential bus master devices that the processor will release bus control at the end of the current bus cycle.

Bus Grant Acknowledge ($\overline{BGACK}$) — This input indicates that some other device has become the bus master. This signal should not be asserted until the following four conditions are met:

1. a bus grant has been received,
2. address strobe is inactive which indicates that the microprocessor is not using the bus,
3. data transfer acknowledge is inactive which indicates that neither memory or peripherals are using the bus, and
4. bus grant acknowledge is inactive which indicates that no other device is still claiming bus mastership.

Table 13. DATA STROBE CONTROL OF DATA BUS

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8 – D15	D0 – D7
High	High	–	No valid data	No valid data
Low	Low	High	Valid data bits 8 – 15	Valid data bits 0 – 7
High	Low	High	No valid data	Valid data bits 0 – 7
Low	High	High	Valid data bits 8 – 15	No valid data
Low	Low	Low	Valid data bits 8 – 15	Valid data bits 0 – 7
High	Low	Low	Valid data bits 0 – 7*	Valid data bits 0 – 7
Low	High	Low	Valid data bits 8 – 15	Valid data bits 8 – 15*

*These conditions are a result of current implementation and may not appear on future devices.

Interrupt Control ($\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$)
These input pins indicate the encoded priority level of the device requesting an interrupt. Level seven is the highest priority while level zero indicates that no interrupts are requested. Level seven cannot be masked. The least significant bit is $\overline{IPL0}$ and the most significant bit is $\overline{IPL2}$. These lines must remain stable until the processor signals interrupt acknowledge (FC0 – FC2 are all high, A4 – A23 are all high) to insure that the interrupt is recognized.

System Control
The system control inputs are used to either reset or halt the processor and to indicate to the processor that bus errors have occurred. The three system control inputs are explained in the following paragraphs.

Bus Error ($\overline{BERR}$) — This input informs the processor that there is a problem with the cycle currently being executed. Problems may be a result of:

1. nonresponding devices,
2. interrupt vector number acquisition failure,
3. illegal access request as determined by a memory management unit, or
4. other application dependent errors.

The bus error signal interacts with the halt signal to determine if the current bus cycle should be reexecuted or if exception processing should be performed.

Refer to **Bus Error and Halt Operation** for additional information about the interaction of the $\overline{BERR}$ and $\overline{HALT}$ signals.

Reset ($\overline{RESET}$) — This bidirectional signal line acts to reset (start a system initialization sequence) the processor in response to an external reset signal. An internally generated reset (result of a reset instruction) causes all external devices to be reset and the internal state of the processor is not affected. A total system reset (processor and external devices) is the result of external $\overline{HALT}$ and $\overline{RESET}$ signals applied at the same time. Refer to **Reset Operation** for further information.

Halt ($\overline{HALT}$) — when this bidirectional line is driven by an external device, it will cause the processor to stop at the completion of the current bus cycle. When the processor has been halted using this input, all control signals are inactive and all three-state lines are put in their high-impedance state (refer to table 15). Refer to **Bus Error and Halt Operation** for additional information about the interaction between the $\overline{HALT}$ and $\overline{BERR}$ signals.

When the processor has stopped executing instructions, due to a double bus fault condition (refer to **Double Bus Faults**), the $\overline{HALT}$ line is driven by the processor to indicate to external devices that the processor has stopped.

Peripheral Control
These control signals are used to allow the interfacing of synchronous peripheral devices with the asynchronous SCN68010. These signals are explained in the following paragraphs.

Enable (E) — This signal is the standard enable signal common to all synchronous type peripheral devices. The period for this output is ten SCN68010 clock periods (six

Preliminary

clocks low, four clocks high). Enable is generated by an internal ring counter which may come up in any state (i.e., at power on, it is impossible to guarantee phase relationship of E to CLK). E is a free-running clock and runs regardless of the state of the bus on the MPU.

Valid Peripheral Address ($\overline{VPA}$) — This input indicates that the device addressed is a synchronous family device and that data transfer should be synchronized with the enable (E) signal. This input also indicates that the processor should use automatic vectoring for an interrupt. Refer to **Interface with Synchronous Peripherals.**

Valid Memory Address ($\overline{VMA}$) — This output is used to indicate to synchronous peripheral devices that there is a valid address on the address bus and the processor is synchronized to enable (E). This signal only responds to a valid peripheral address ($\overline{VPA}$) input which indicates that the peripheral is a synchronous family device.

Processor Status (FC0, FC1, FC2).
These function code outputs indicate the state (user or supervisor) and the cycle type currently being executed, as shown in table 14. The information indicated by the function code outputs is valid whenever address strobe ($\overline{AS}$) is active.

Clock (CLK)
The clock input is a TTL-compatible signal that is internally buffered for development of the internal clocks needed by the processor. The clock input should not be gated off at any time and the clock signal must conform to minimum and maximum pulse width times.

Signal Summary
Table 15 is a summary of all the signals discussed in the previous paragraphs.

Table 14. FUNCTION CODE OUTPUTS

FUNCTION CODE OUTPUT			CYCLE TYPE
FC2	FC1	FC0	
Low	Low	Low	(Undefined, reserved)
Low	Low	High	User data
Low	High	Low	User program
Low	High	High	(Undefined, reserved)
High	Low	Low	(Undefined, reserved)
High	Low	High	Supervisor data
High	High	Low	Supervisor program
High	High	High	CPU space

Table 15. SIGNAL SUMMARY

SIGNAL NAME	MNEMONIC	INPUT/OUTPUT	ACTIVE STATE	Hi-Z	
				on $\overline{HALT}$	on $\overline{BGACK}$
Address bus	A1 – A23	Output	High	Yes	Yes
Data bus	D0 – D15	Input/output	High	Yes	Yes
Address strobe	$\overline{AS}$	Output	Low	No	Yes
Read/write	R/$\overline{W}$	Output	Read-High Write-Low	No No	Yes Yes
Upper and lower data strobes	$\overline{UDS}$, $\overline{LDS}$	Output	Low	No	Yes
Data transfer acknowledge	$\overline{DTACK}$	Input	Low	–	–
Bus request	$\overline{BR}$	Input	Low	–	–
Bus grant	$\overline{BG}$	Output	Low	No	No
Bus grant acknowledge	$\overline{BGACK}$	Input	Low	–	–
Interrupt priority level	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	Input	Low	–	–
Bus error	$\overline{BERR}$	Input	Low	–	–
Reset	$\overline{RESET}$	Input/Output	Low	No*	No*
Halt	$\overline{HALT}$	Input/Output	Low	No*	No*
Enable	E	Output	High	No	No
Valid memory address	$\overline{VMA}$	Output	Low	No	Yes
Valid peripheral address	$\overline{VPA}$	Input	Low	–	–
Function code output	FC0, FC1, FC2	Output	High	No	Yes
Clock	CLK	Input	High	–	–
Power input	V_{CC}	Input	–	–	–
Ground	GND	Input	–	–	–

*Open drain

Preliminary

Bus Operation

The following paragraphs explain control signal and bus operation during data transfer operations, bus arbitration, bus error and halt conditions, and reset operation.

Data Transfer Operations

Transfer of data between devices involves the following signals:

1. address bus A1 through A23,
2. data bus D0 through D15, and
3. control signals.

The address and data buses are separate parallel buses used to transfer data using an asynchronous bus structure. In all cycles, the bus master assumes responsibility for deskewing all signals it issues at both the start and end of a cycle. In addition, the bus master is responsible for deskewing the acknowledge and data signals from the slave device.

The following paragraphs explain the read, write, and read-modify-write cycles. The indivisible read-modify-write cycle is the method used by the SCN68010 for interlocked multiprocessor communications.

Read Cycle — During a read cycle, the processor receives data from the memory or a peripheral device. The processor reads bytes of data in all cases. If the instruction specifies a word (or long word) operation, the processor reads both upper and lower bytes simultaneously by asserting both upper and lower data strobes. When the instruction specifies byte operation, the processor uses an internal A0 bit to determine which byte to read and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. When the data is received, the processor correctly positions it internally. If $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ is not asserted for the required set-up time before the falling edge of S4, a wait cycle will be inserted in the bus cycle and $\overline{DTACK}$ will be sampled again on the falling edge of each wait cycle. The SCN68010 will continue to insert wait cycles until $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ is recognized.

A word read cycle flowchart is given in figure 9. A byte read cycle flowchart is given in figure 10. Read cycle timing is given in figure 11. Figure 12 details word and byte read cycle operations.

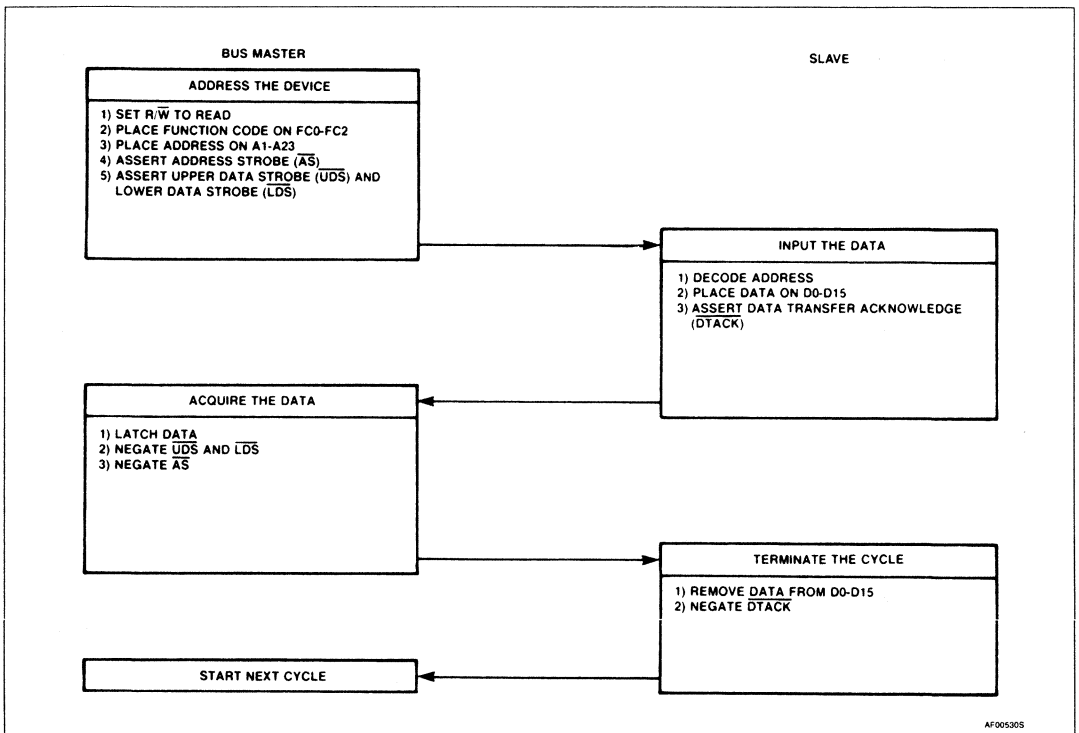


Figure 9. Word Read Cycle Flowchart

Preliminary

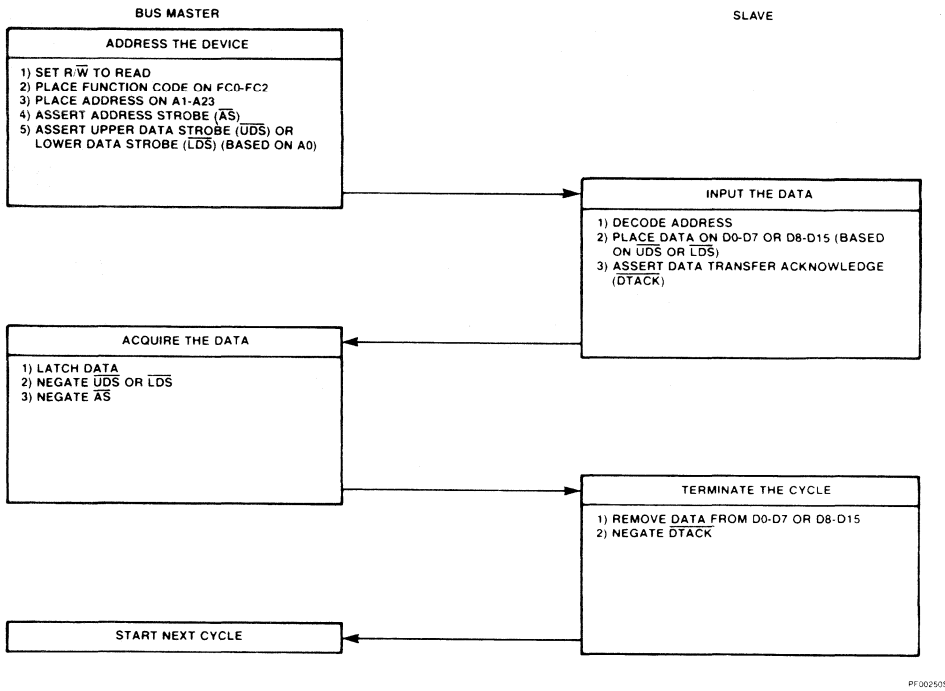


Figure 10. Byte Read Cycle Flowchart

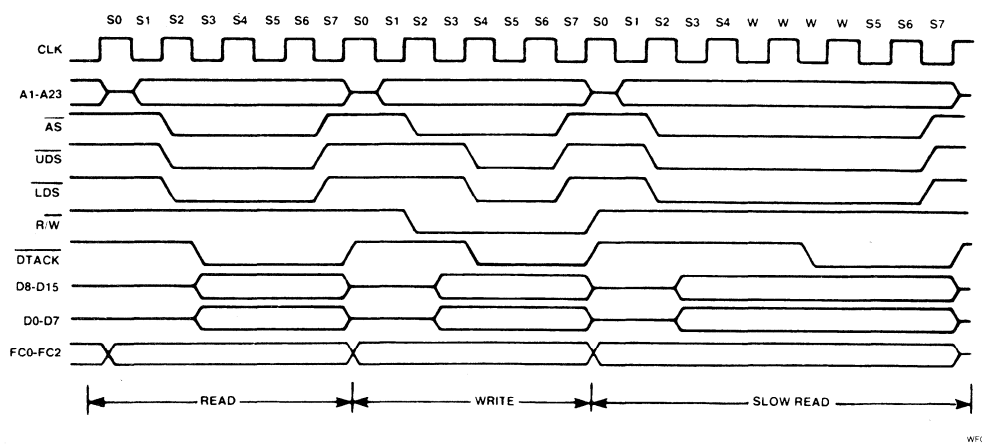


Figure 11. Read and Write Cycle Timing Diagram

Preliminary

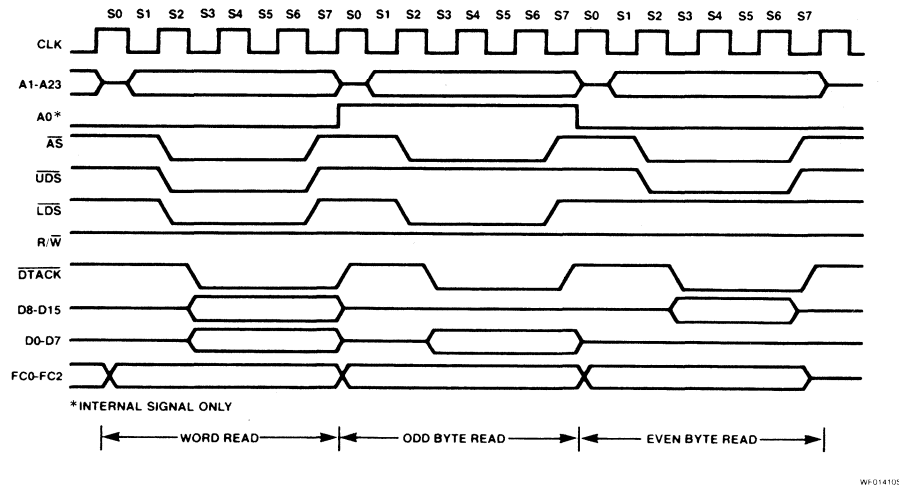


Figure 12. Word and Byte Read Cycle Timing Diagram

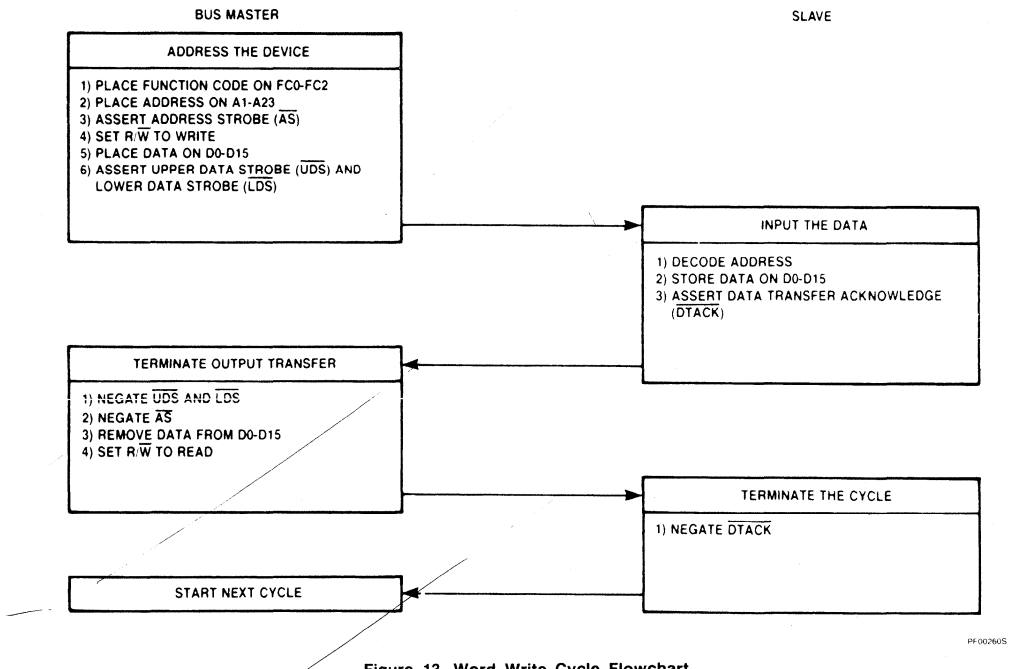


Figure 13. Word Write Cycle Flowchart

Preliminary

Write Cycle — During a write cycle, the processor sends data to either the memory or a peripheral device. The processor writes bytes of data in all cases. If the instruction specifies a word operation, the processor writes both bytes. When the instruction specifies a byte operation, the processor uses an internal A0 bit to determine which byte to write and then issues the data strobe required for that byte. For byte operations, when the A0 bit equals zero, the upper data strobe is issued. When the A0 bit equals one, the lower data strobe is issued. A word write flowchart is given in figure 13. A byte write cycle flowchart is given in figure 14. Write cycle timing is given in Figure 11. Figure 15 details word and byte write cycle operation.

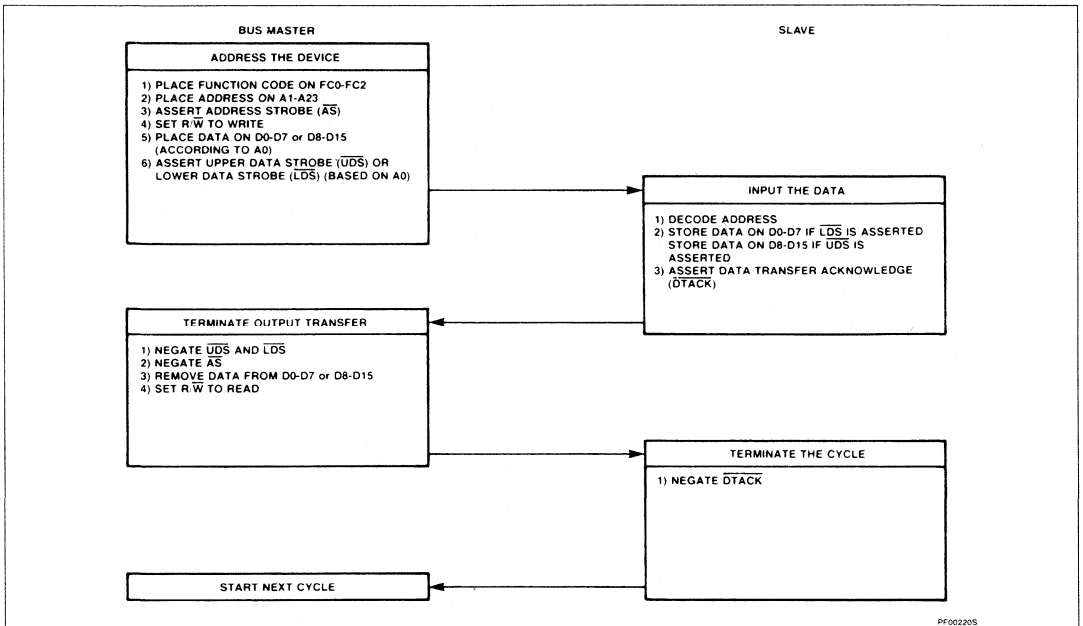


Figure 14. Byte Write Cycle Flowchart

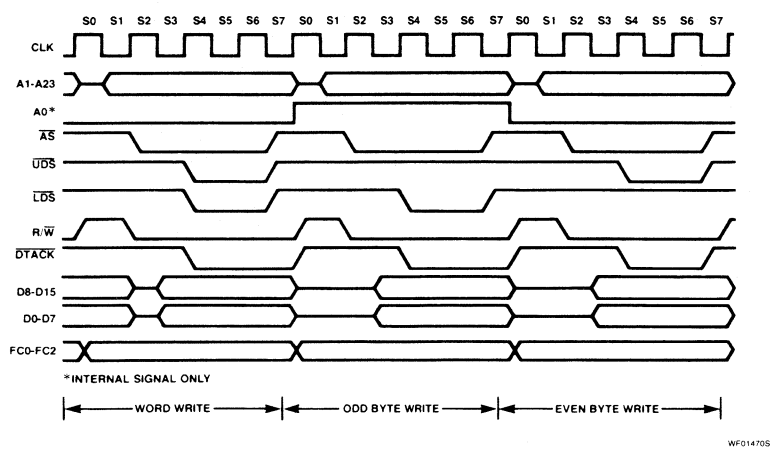


Figure 15. Word and Byte Write Cycle Timing Diagram

Preliminary

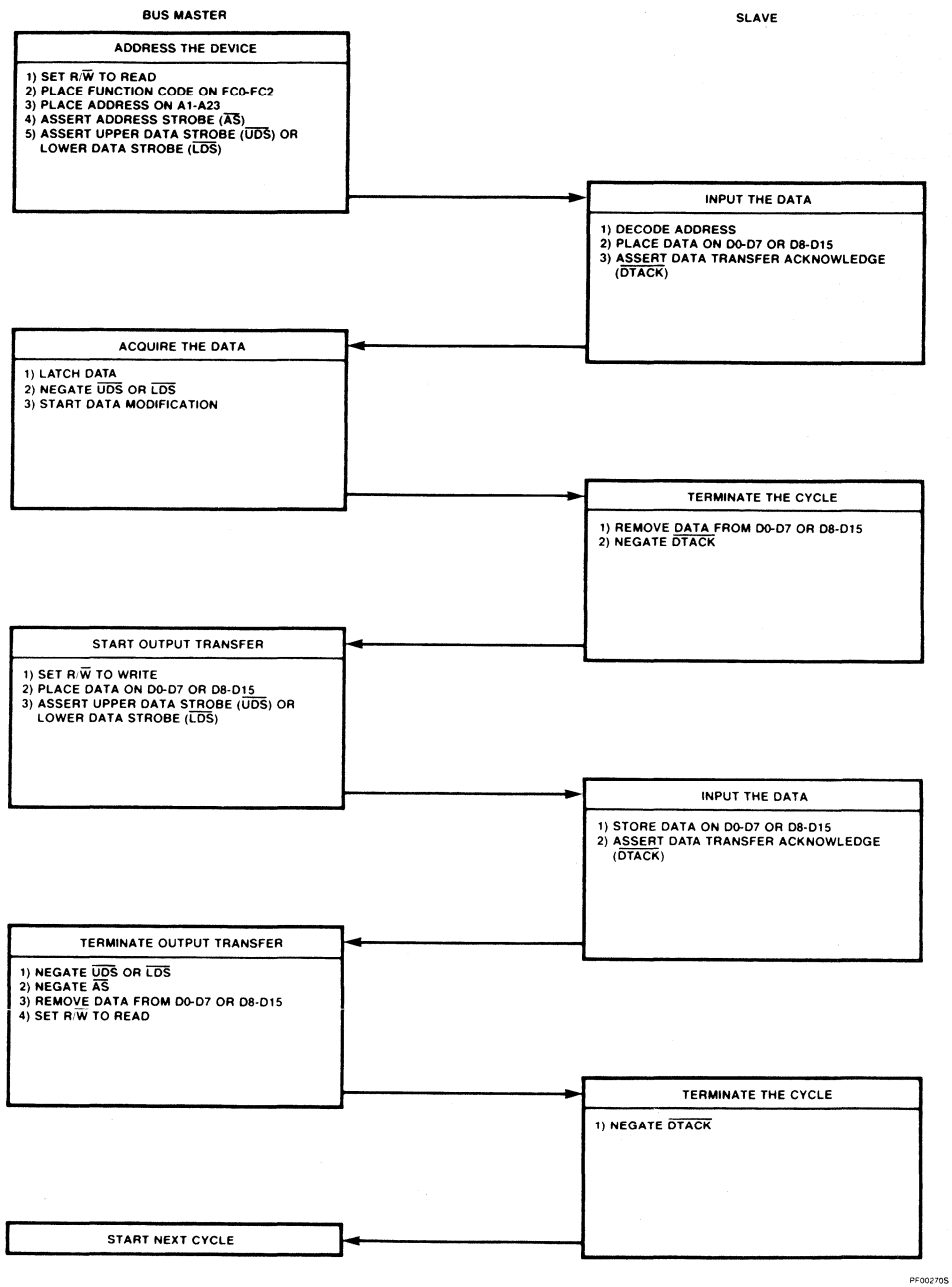


Figure 16. Read-Modify-Write Cycle Flowchart

Preliminary

Read-Modify-Write Cycle — The read-modify-write cycle performs a read, modifies the data in the arithmetic-logic unit, and writes the data back to the same address. In the SCN68010, this cycle is indivisible in that the address strobe is asserted throughout the entire cycle. The test and set (TAS) instruction uses this cycle to provide meaningful communication between processors in a multiple processor environment. This instruction is the only instruction that uses the read-modify-write cycle and since the test and set instruction only operates on bytes, all read-modify-write cycles are byte operations. A read-modify-write flowchart is given in figure 16 and a timing diagram is given in figure 17.

Wait cycles will be inserted between S4 and S5 on the read portion of the bus cycle and between S16 and S17 on the write portion of the cycle if $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ is not asserted for the required set-up time prior to the falling edge of S4 and S16 respectively.

CPU Space Cycle

During a CPU space cycle, the SCN68010 reads a peripheral device vector number or indicates a breakpoint instruction. If the cycle is to read a vector number it is referred to as an interrupt acknowledge cycle. A CPU space cycle is indicated when the function codes are all high. The address bus then defines what type of CPU space cycle is being executed. The SCN68010 defines two types of CPU space cycles, the interrupt acknowledge cycle, and the breakpoint cycle.

The interrupt acknowledge cycle on an S68000 family compatible processor is defined as a CPU space cycle with the most significant address lines high; on the SCN68010 this means that A4 – A23 will be high. The level of the interrupt being acknowledged is encoded on address lines A1 – A3. An interrupt acknowledge cycle is terminated in the same manner as a normal read cycle. The processor expects a peripheral device to respond to an interrupt acknowledge cycle with a vector number that will be used to transfer control to an interrupt handler routine. See **Interrupts** for further discussion of the interrupt acknowledge cycle.

The breakpoint read cycle is executed by the SCN68010 in response to a breakpoint illegal instruction. A breakpoint cycle on the SCN68010 is defined as a CPU space cycle with all of the address lines low. The processor does not accept or send any data during this cycle. The breakpoint cycle may be terminated by $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$. See **Illegal and Unimplemented Instructions** for further discussion of breakpoints.

In order to maintain compatibility with future processor members of the S68000 family, system designers should fully decode the CPU space. In particular, the most significant address lines (A4 – A19 for the SCN68008 and A4 – A23 for the SCN68000 or SCN68010) should be used to distinguish between an interrupt acknowledge cycle and a breakpoint cycle. All encodings of bits A16 – A19 are reserved by Signetics for future extensions of the CPU space functions.

Bus Arbitration

Bus arbitration is a technique used by master-type devices to request, be granted, and acknowledge bus mastership. In its simplest form, it consists of the following:

1. asserting a bus mastership request,
2. receiving a grant that the bus is available at the end of the current cycle, and
3. acknowledging that mastership has been assumed.

Figure 18 is a flowchart showing the detail involved in a request from a single device. Figure 19 is a timing diagram for the same operation. This technique allows processing of bus requests during data transfer cycles.

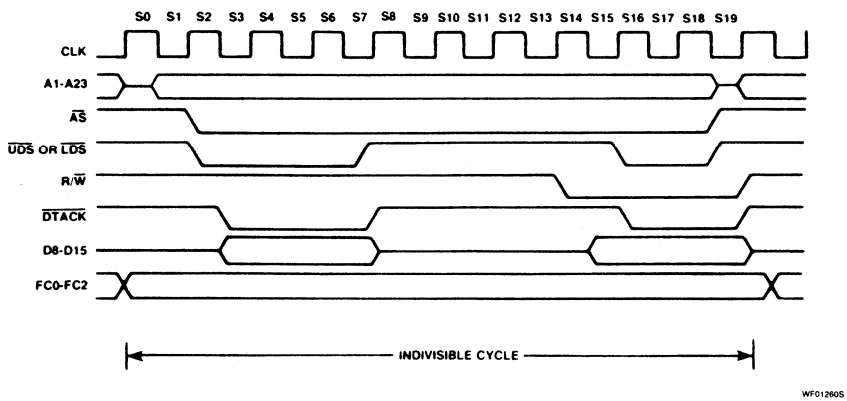


Figure 17. Read-Modify-Write Cycle Timing Diagram

Preliminary

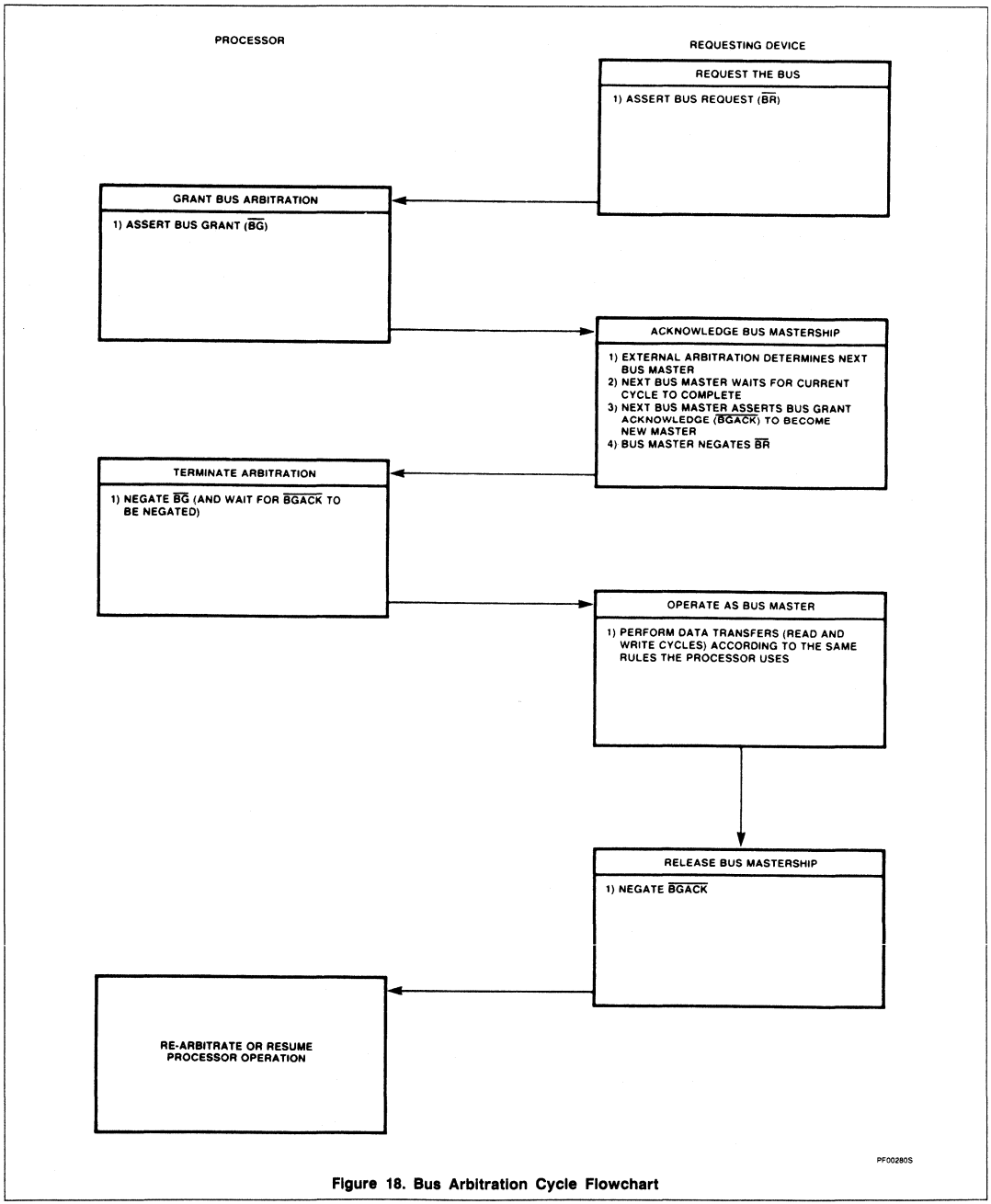


Figure 18. Bus Arbitration Cycle Flowchart

Preliminary

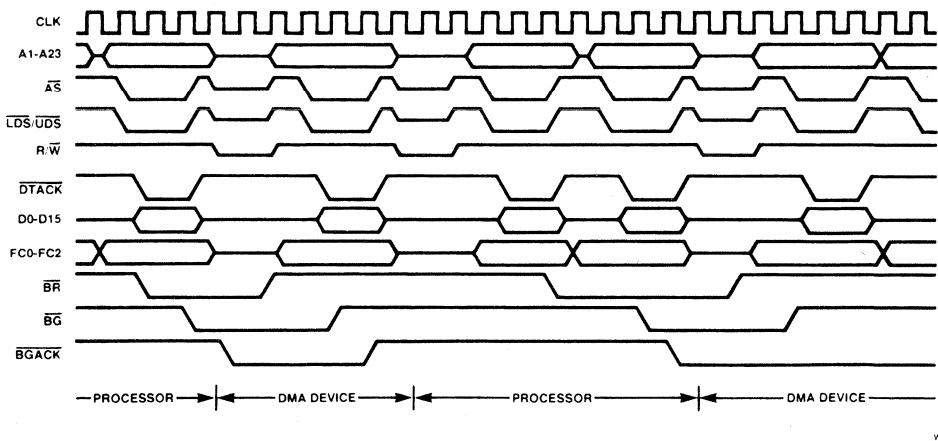


Figure 19. Bus Arbitration Cycle Timing Diagram

The timing diagram shows that the bus request is negated at the time that an acknowledge is asserted. This type of operation would be true for a system consisting of the processor and one device capable of bus mastership. In systems having a number of devices capable of bus mastership, the bus request line from each device is wire ORed to the processor. In this system, it is easy to see that there could be more than one bus request being made. The timing diagram shows that the bus grant signal is negated a few clock cycles after the transition of the acknowledge ($\overline{BGACK}$) signal.

However, if bus requests are still pending, the processor will assert another bus grant within a few clock cycles after it was negated. This additional assertion of bus grant allows external arbitration circuitry to select the next bus master before the current bus master has completed its requirements. The following paragraphs provide additional information about the three steps in the arbitration process.

Requesting the Bus — External devices capable of becoming bus masters request the bus by asserting the bus request ($\overline{BR}$) signal. This is a wire-ORed signal (although it need not be constructed from open-collector devices) that indicates to the processor that some external device requires control of the external bus. The processor is effectively at a lower bus priority level than the external device and will relinquish the bus after it has completed the last bus cycle it has started.

When no acknowledge is received before the bus request signal goes inactive, the processor will continue processing when it detects that the bus request is inactive. This allows ordinary processing to continue if the arbitration circuitry responded to noise inadvertently.

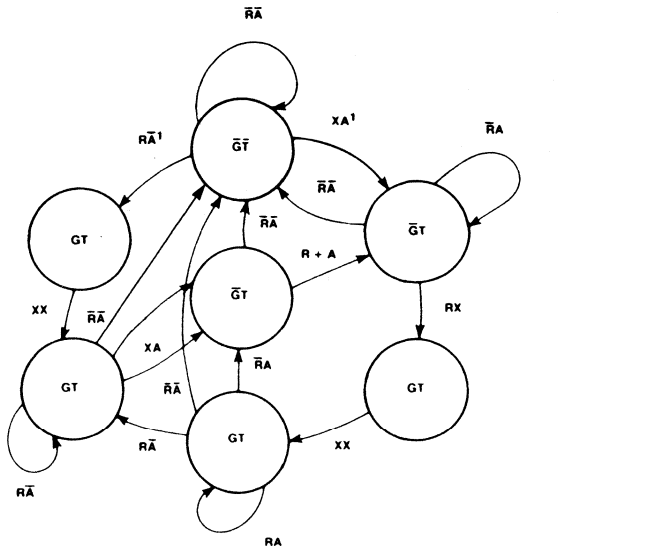
Receiving the Bus Grant — The processor asserts bus grant ($\overline{BG}$) as soon as possible. Normally this is immediately after internal synchronization. The only exception to this occurs when the processor has made an internal decision to execute the next bus cycle but has not progressed far enough into the cycle to have asserted the address strobe ($\overline{AS}$) signal. In this case, bus grant will be delayed until $\overline{AS}$ is asserted to indicate to external devices that a bus cycle is being executed.

The bus grant signal may be routed through a daisy-chained network or through a specific priority-encoded network. The processor is not affected by the external method of arbitration as long as the protocol is obeyed.

Acknowledgement of Mastership — Upon receiving a bus grant, the requesting device waits until address strobe, data transfer acknowledge, and bus grant acknowledge are negated before issuing its own $\overline{BGACK}$. The negation of the $\overline{AS}$ indicates that the previous master has completed its cycle; the negation of $\overline{BGACK}$ indicates that the previous master has released the bus. (While address strobe is asserted, no device is allowed to "break into" a cycle.) The negation of $\overline{DTACK}$ indicates the previous slave has terminated its connection to the previous master. Note that in some applications data transfer acknowledge might not enter into this function. General purpose devices would then be connected such that they were only dependent on address strobe. When bus grant acknowledge is issued, the device is a bus master until it negates bus grant acknowledge. Bus grant acknowledge should not be negated until after the bus cycle(s) is (are) completed. Bus mastership is terminated at the negation of bus grant acknowledge.

The bus request from the granted device should be negated after bus grant acknowledge is asserted. If a bus request is still pending, another bus grant will be asserted within a few clocks of the negation of the bus grant. Refer to **Bus Arbitration Control.** Note that the processor does not perform any external bus cycles before it re-asserts bus grant.

Preliminary



NOTES:
R = Bus request internal
A = Bus grant acknowledge internal
G = Bus grant
T = Three-state control to bus control logic[2]
X = Don't care
1. State machine will not change if the bus is S0 or S1. Refer to Bus Arbitration Control.
2. The address bus will be placed in the high-impedance state if T is asserted and $\overline{AS}$ is negated.

Figure 20. SCN68010 Bus Arbitration Unit State Diagram

Bus Arbitration Control

The bus arbitration control unit in the SCN68010 is implemented with a finite state machine. A state diagram of this machine is shown in figure 20. All asynchronous signals to the SCN68010 are synchronized before they are used internally. This synchronization is accomplished in a maximum of one cycle of the system clock, assuming that the asynchronous input set-up time (#47) has been met (see figure 21). The input signal is sampled on the falling edge of the clock and is valid internally after the next rising edge.

As shown in figure 20, input signals labeled R and A are internally synchronized on the bus request and bus grant acknowledge pins respectively. The bus grant output is labeled G and the internal three-state control signal T. If T is true, the address, data, and control buses are placed in a high-impedance state when $\overline{AS}$ is negated. All signals are shown in positive logic (active high) regardless of their true active voltage level.

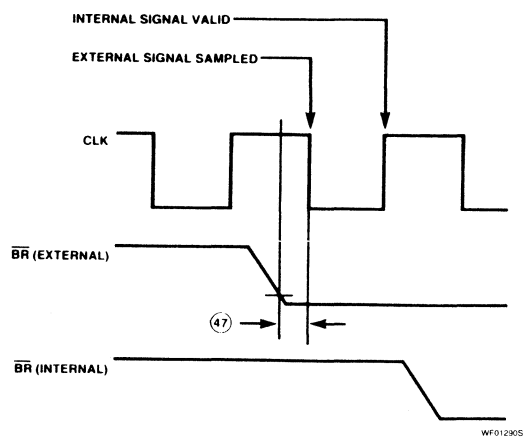


Figure 21. Timing Relationship of External Asynchronous Inputs to Internal Signals

Preliminary

State changes (valid outputs) occur on the next rising edge after the internal signal is valid.

A timing diagram of the bus arbitration sequence during a processor bus cycle is shown in figure 22. The bus arbitration sequence while the bus is inactive (i.e., executing internal operations such as a multiply instruction) is shown in figure 23.

If a bus request is made at a time when the MPU has already begun a bus cycle but $\overline{AS}$ has not been asserted (bus state S0), $\overline{BG}$ will not be asserted on the next rising edge. Instead, $\overline{BG}$ will be delayed until the second rising edge following its internal assertion. This sequence is shown in figure 24.

Bus Error and Halt Operation

In a bus architecture that requires a handshake from an external device, the possibility exists that the handshake might not occur. Since different systems will require a different maximum response time, a bus error input is provided. External circuitry must be used to determine the duration between address strobe and data transfer acknowledge before issuing a bus error signal. When a bus error or/and halt signal is received, the processor will initiate a bus error exception sequence or try to re-run the bus cycle.

In addition to a bus timeout indicator, the bus error input is used to indicate a page fault in a virtual memory system. When an external memory management unit detects an invalid access, a bus error is signaled to suspend execution of the current instruction.

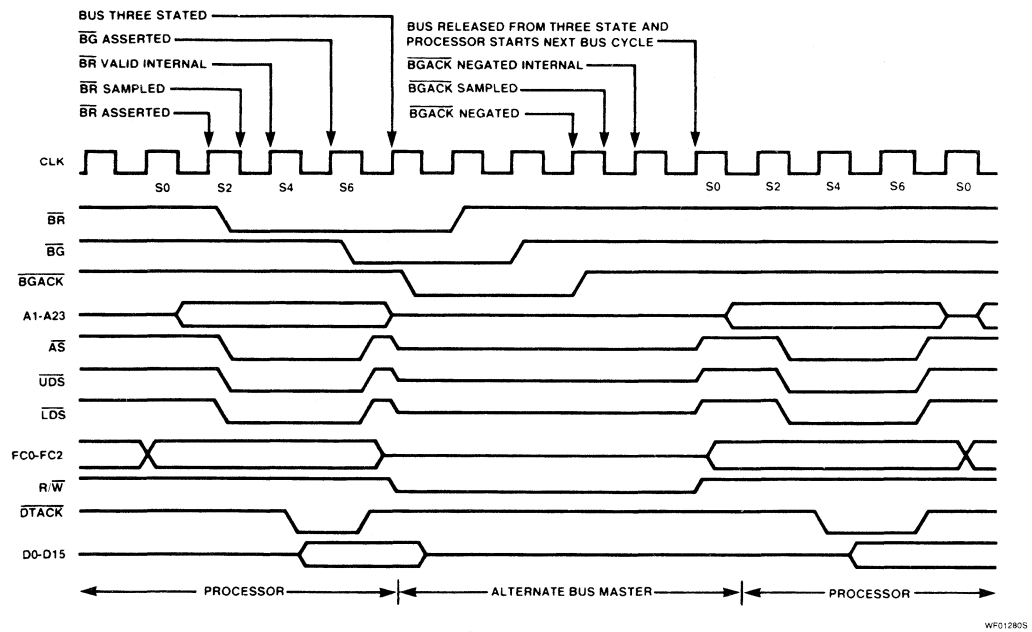


Figure 22. Bus Arbitration Timing Diagram — Processor Active

Preliminary

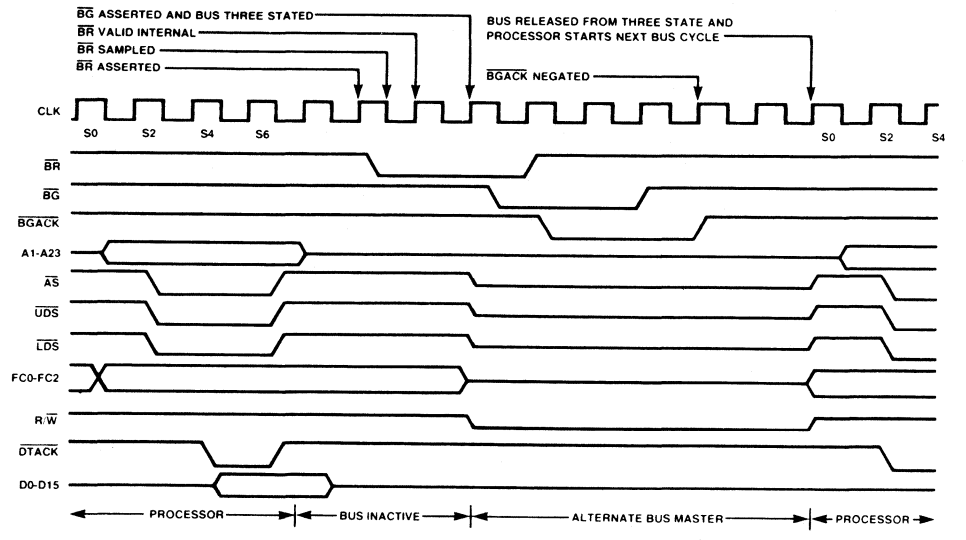


Figure 23. Bus Arbitration Timing Diagram — Bus Inactive

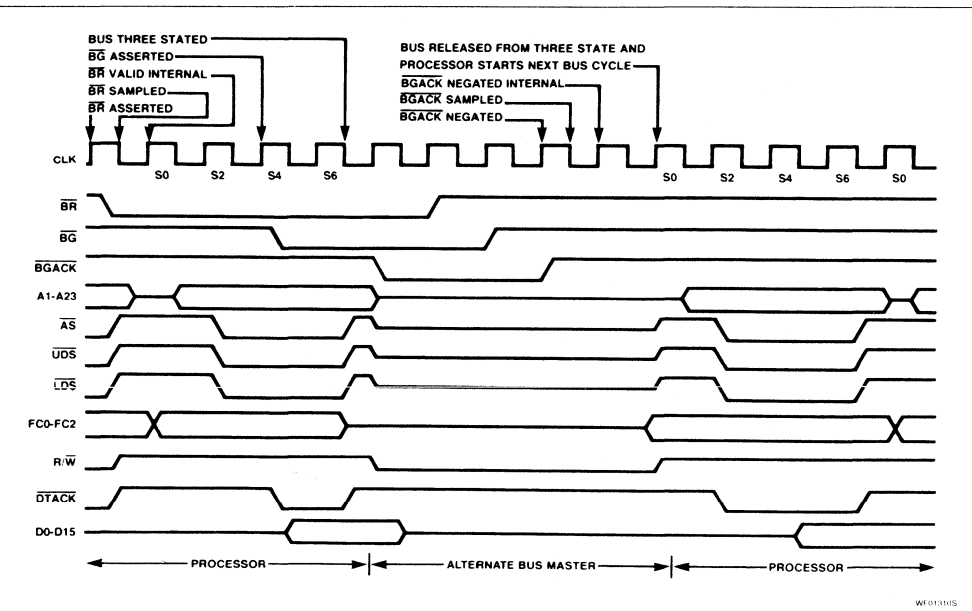


Figure 24. Bus Arbitration Timing Diagram — Special Case

Preliminary

Bus Error Operation — When the bus error signal is used to terminate a bus cycle, the SCN68010 will enter exception processing immediately following the bus cycle. The bus error signal is recognized in either of the following cases:

1. $\overline{DTACK}$ and $\overline{HALT}$ are negated and $\overline{BERR}$ is asserted.
2. $\overline{HALT}$ and $\overline{BERR}$ are negated and $\overline{DTACK}$ is asserted. $\overline{BERR}$ is then asserted within one clock cycle.

When the bus error condition is recognized, the current bus cycle will be terminated in S9 for a read cycle, a write cycle, or the read portion of a read-modify-write cycle and in S21 of the write portion of a read-modify-write cycle. As long as $\overline{BERR}$ remains asserted, the data and address buses will be in the high-impedance state. Figures 25 and 26 show the timing diagrams for both types of bus error signals.

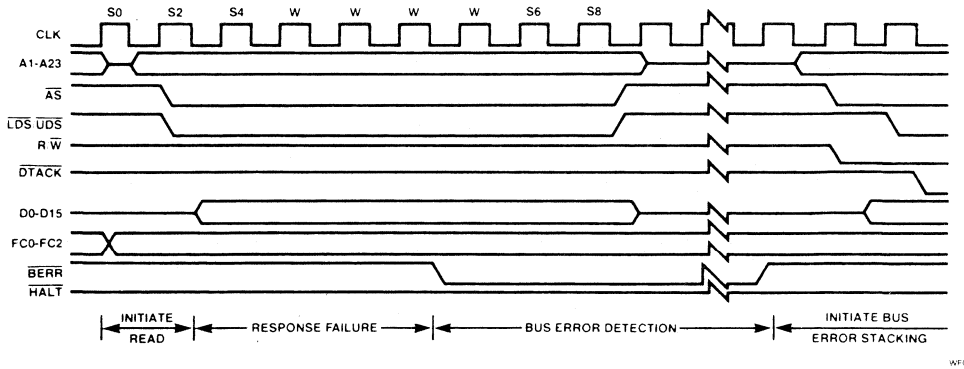


Figure 25. Bus Error Timing Diagram

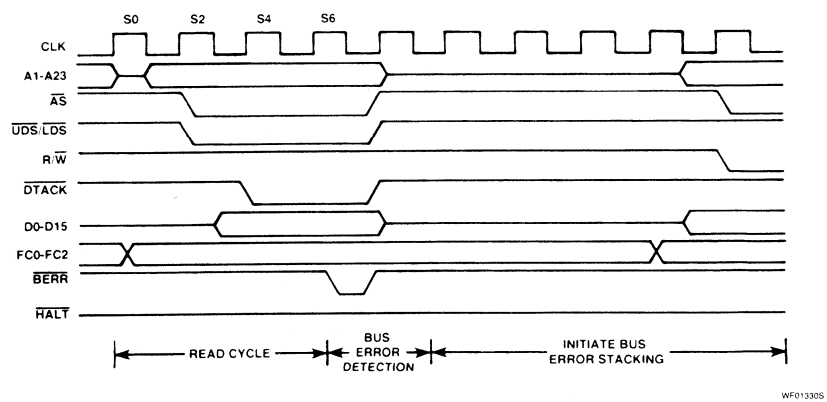


Figure 26. Delayed Bus Error Timing Diagram

Preliminary

After the aborted bus cycle is terminated and $\overline{BERR}$ is negated, the SCN68010 enters exception processing for the bus error exception. During the exception processing sequence, the following information is placed on the supervisor stack:

1. Status register
2. Program counter (two words, may be up to five words past the instruction being executed)
3. Frame format and vector offset
4. Internal register information, 22 words

Note that the first four words of information are identical to the information stacked by any other exception such as an interrupt or TRAP instruction. The additional information is used by the SCN68010 to continue the execution of the suspended instruction when it is reloaded by an RTE instruction. See **Bus Error** for further details.

After the SCN68010 has placed the above information on the stack, the bus error exception vector is read from vector table entry number two (offset $08) and placed in the program counter. The processor then resumes instruction execution.

NOTE

If a read-modify-write instruction is terminated with a bus error and later continued with an RTE instruction, the processor will re-run the entire cycle whether the bus error occurred on the read or the write portion of the cycle.

Re-Run Operation — When, during a bus cycle, the processor receives a bus error signal and the halt pin is being driven by an external device, the processor enters the re-run sequence. A delayed re-run signal may be used similarly to the delayed bus error signal described above. Figures 27 and 28 are timing diagrams for both methods of re-running the bus cycle.

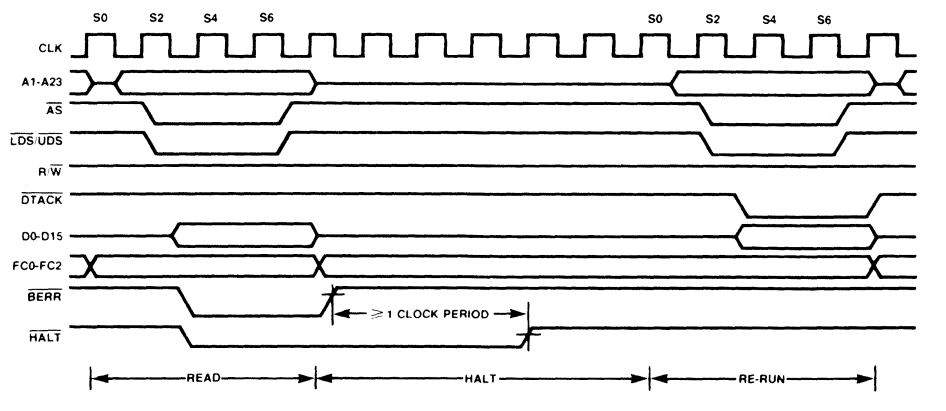


Figure 27. Re-Run Bus Cycle Timing Diagram

Preliminary

The processor terminates the bus cycle, then puts the address and data lines in the high-impedance state. The processor remains "halted", and will not run another bus cycle until the halt signal is removed by external logic. Then the processor will re-run the previous cycle using the same function codes, the same data (for a write operation), and the same address. The bus error signal should be removed at least one clock cycle before the halt signal is removed.

NOTE
The processor will not re-run a read-modify-write cycle. This restriction is made to guarantee that the entire cycle runs correctly and that the write operation of a test-and-set operation is performed without ever releasing $\overline{AS}$. If $\overline{BERR}$ and $\overline{HALT}$ are asserted during a read-modify-write bus cycle, a bus error operation results.

Halt Operation — The halt input signal to the SCN68010 performs a halt/run/single-step function in a similar fashion to the synchronous halt function. The halt and run modes are somewhat self explanatory in that when the halt signal is constantly active the processor "halts" (does nothing) and when the halt signal is constantly inactive the processor "runs" (does something).

This single-step mode is derived from correctly timed transitions on the halt signal input. It forces the processor to execute a single bus cycle by entering the run mode until the processor starts a bus cycle then changing to the halt mode. Thus, the single-step mode allows the user to proceed through (and therefore debug) processor operations one bus cycle at a time.

Figure 29 details the timing required for correct single-step operations. Some care must be exercised to avoid harmful interactions between the bus error signal and the halt pin when using the single-cycle mode as a debugging tool. This is also true of interactions between the halt and reset lines since these can reset the machine.

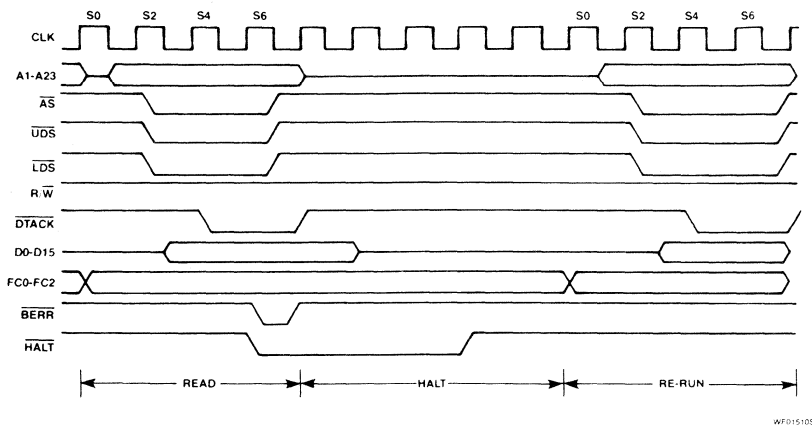


Figure 28. Delayed Re-Run Bus Cycle Timing Diagram

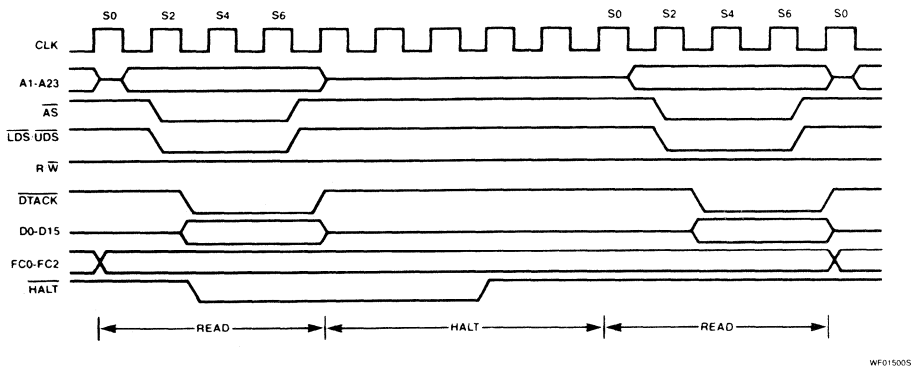


Figure 29. Halt Processor Timing Diagram

Preliminary

When the processor completes a bus cycle after recognizing that the halt signal is active, most three-state signals are put in the high-impedance state, these include:

1. address lines, and
2. data lines.

This is required for correct performance of the re-run bus cycle operation.

While the processor is honoring the halt request, bus arbitration performs as usual. That is, halting has no effect on bus arbitration. It is the bus arbitration function that removes the control signals from the bus.

The halt function and the hardware trace capability allow the hardware debugger to trace single bus cycles or single instructions at a time. These processor capabilities, along with a software debugging package, give total debugging flexibility.

Double Bus Faults — When a bus error exception occurs, the processor will attempt to stack several words containing information about the state of the machine. If a bus error exception occurs during the stacking operation, there have been two bus errors in a row. This is commonly referred to as a double bus fault. When a double bus fault occurs, the processor will halt and drive the $\overline{HALT}$ line low. Once a bus error exception has occurred, any bus error exception occurring before the execution of the next instruction constitutes a double bus fault.

Note that a bus cycle which is re-run does not constitute a bus error exception and does not contribute to a double bus fault. Note also that this means that as long as the external hardware requests it, the processor will continue to re-run the same bus cycle.

The bus error pin also has an effect on processor operation after the processor receives an external reset input. The processor reads the vector table after a reset to determine the address to start program execution. If a bus error occurs while reading the vector table (or at any time before the first instruction is executed), the processor reacts as if a double bus fault has occurred and it halts. Only an external reset will start a halted processor.

Reset Operation

The reset signal is a bidirectional signal that allows either the processor or an external device to reset the system. Figure 30 is a timing diagram for the reset operation. Both the halt and reset lines must be asserted to ensure total reset of the processor in all cases.

When the reset and halt lines are driven by an external device, it is recognized as an entire system reset, including the processor. The processor responds by reading the reset vector table entry (vector number zero, address $000000) and loads it into the supervisor stack pointer (SSP). Vector table entry number one at address $000004 is read next and loaded into the program counter. The processor initializes the status register to an interrupt level of seven and the vector base register to $00000000. No other registers are affected by the reset sequence.

When a reset instruction is executed, the processor drives the reset pin for 124 clock periods. In this case, the processor is trying to reset the rest of the system. Therefore, there is no effect on the internal state of the processor. All of the processor's internal registers and the status register are unaffected by the execution of a reset instruction. All external devices connected to the reset line should be reset at the completion of the reset instruction.

Asserting the $\overline{RESET}$ and $\overline{HALT}$ lines for ten clock cycles will cause a processor reset, except when V_{CC} is initially applied to the processor. In this case, an external reset must be applied for at least 100 milliseconds.

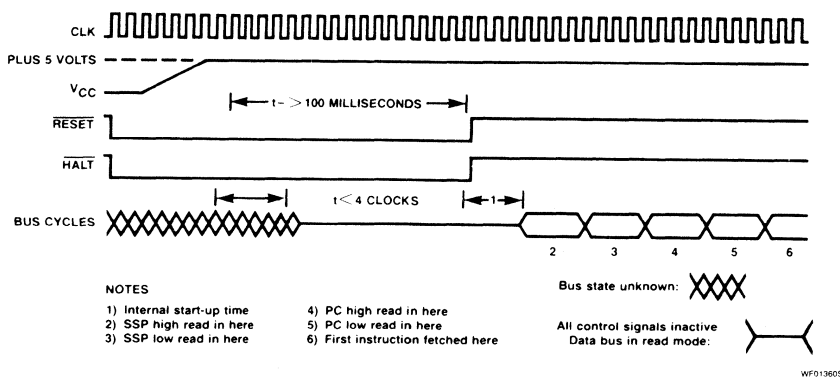


Figure 30. Reset Operation Timing Diagram

Preliminary

The Relationship of $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$

In order to properly control termination of a bus cycle for a re-run or a bus error condition, $\overline{DTACK}$, $\overline{BERR}$, and $\overline{HALT}$ should be asserted and negated on the rising edge of the SCN68010 clock. This will assure that when two signals are asserted simultaneously, the required set-up time (#47) for both of them will be met during the same bus state. This, or some equivalent precaution, should be designed external to the SCN68010.

The preferred bus cycle terminations may be summarized as follows (case numbers refer to table 16):

Normal Termination:
$\overline{DTACK}$ is asserted, $\overline{BERR}$ and $\overline{HALT}$ remain negated (case 1).

Halt Termination:
$\overline{HALT}$ is asserted at same time, or before $\overline{DTACK}$ and $\overline{BERR}$ remains negated (case 2).

Bus Error Termination:
$\overline{BERR}$ is asserted in lieu of, at the same time, or before $\overline{DTACK}$ (case 3) or after $\overline{DTACK}$ (case 4) and $\overline{HALT}$ remains negated; $\overline{BERR}$ is negated at the same time or after $\overline{DTACK}$.

Re-Run Termination:
$\overline{HALT}$ and $\overline{BERR}$ are asserted in lieu of, at the same time, or before $\overline{DTACK}$ (case 5) or after $\overline{DTACK}$ (case 6); $\overline{BERR}$ is negated at the same time or after $\overline{DTACK}$, $\overline{HALT}$ must be held at least one cycle after $\overline{BERR}$.

Table 16 details the resulting bus cycle termination under various combinations of control signal sequences. The negation of these same control signals under several conditions is shown in table 17. ($\overline{DTACK}$ is assumed to be negated normally in all cases; for best results, both $\overline{DTACK}$ and $\overline{BERR}$ should be negated when address strobe is negated).

EXAMPLE A:
A system uses a watch-dog timer to terminate accesses to unpopulated address space. The timer asserts $\overline{BERR}$ after time out (case 3).

EXAMPLE B:
A system uses error detection on RAM contents. Designer may:

a) Delay $\overline{DTACK}$ until data verified, and return $\overline{BERR}$ and $\overline{HALT}$ simultaneously to re-run error cycle (case 5), or if valid, return $\overline{DTACK}$ (case 1).
b) Delay $\overline{DTACK}$ until data verified, and return $\overline{BERR}$ at same time as $\overline{DTACK}$ if data in error (case 3).
c) Return $\overline{DTACK}$ prior to data verification, as described in the next section. If data is invalid, $\overline{BERR}$ is asserted on next clock cycle (case 4).
d) Return $\overline{DTACK}$ prior to data verification, if data is invalid assert $\overline{BERR}$ and $\overline{HALT}$ on next clock cycle (case 6). The memory controller may then correct the RAM prior to or during the re-run.

Table 16. $\overline{DTACK}$, $\overline{BERR}$, AND $\overline{HALT}$ ASSERTION RESULTS

CASE NO.	CONTROL SIGNAL	ASSERTED ON RISING EDGE OF STATE		RESULT
		N	N + 2	
1	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	S NA X	Normal cycle terminate and continue.
2	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA A/S	S NA S	Normal cycle terminate and halt. Continue when $\overline{HALT}$ removed.
3	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A NA	X S NA	Terminate and take bus error trap.
4	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	X A NA	Terminate and take bus error trap.
5	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	X A A/S	X S S	Terminate and re-run when $\overline{HALT}$ removed.
6	$\overline{DTACK}$ $\overline{BERR}$ $\overline{HALT}$	A NA NA	X A A	Terminate and re-run when $\overline{HALT}$ removed.

NOTES:
N – the number of the current even bus state (e.g., S4, S6, etc.)
A – signal is asserted in this bus state
NA – signal is not asserted in this state
X – don't care
S – signal was asserted in previous state and remains asserted in this state

Table 17. $\overline{BERR}$ AND $\overline{HALT}$ NEGATION RESULTS

CONDITIONS OF TERMINATION IN TABLE 16	CONTROL SIGNAL	NEGATED ON RISING EDGE OF STATE			RESULTS – NEXT CYCLE
		N		N+2	
Bus Error	$\overline{BERR}$ $\overline{HALT}$	• •	or or	• •	Takes bus error trap.
Re-run	$\overline{BERR}$ $\overline{HALT}$	• •	or	•	Illegal sequence; usually traps to vector number 0.
Re-run	$\overline{BERR}$ $\overline{HALT}$	•		• •	Re-runs the bus cycle.
Normal	$\overline{BERR}$ $\overline{HALT}$	• •	or	•	May lengthen next cycle.

• = Signal is negated in this bus state.

Preliminary

Asynchronous Versus Synchronous Operation

Asynchronous Operation

To achieve clock frequency independence at a system level, the SCN68010 can be used in an asynchronous manner. This entails using only the bus handshake lines ($\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{HALT}$ and $\overline{VPA}$) to control the data transfer. Using this method, $\overline{AS}$ signals the start of a bus cycle and the data strobes are used as a condition for valid data on a write cycle. The slave device (memory or peripheral) then responds by placing the requested data on the data bus for a read cycle or latching data on a write cycle and asserting the data transfer acknowledge signal ($\overline{DTACK}$) to terminate the bus cycle. If no slave responds or the access is invalid, external control logic asserts the $\overline{BERR}$, or $\overline{BERR}$ and $\overline{HALT}$, signal to abort or re-run the bus cycle.

The $\overline{DTACK}$ signal is allowed to be asserted before the data from a slave device is valid on a read cycle. The length of time that $\overline{DTACK}$ may precede data is given as parameter #31 (See **AC Electrical Characteristics** for # references) and it must be met in any asynchronous system to insure that valid data is latched into the processor. Notice that there is no maximum time specified from the assertion of $\overline{AS}$ to the assertion of $\overline{DTACK}$. This is because the MPU will insert wait cycles of one clock period each until $\overline{DTACK}$ is recognized.

The $\overline{BERR}$ signal is allowed to be asserted after the $\overline{DTACK}$ signal is asserted. $\overline{BERR}$ must be asserted within the time given as parameter #48 after $\overline{DTACK}$ is asserted in any asynchronous system to insure proper operation. If this maximum delay time is violated, the processor may exhibit erratic behavior.

Synchronous Operation

To allow for those systems which use the system clock as a signal to generate $\overline{DTACK}$ and other asynchronous inputs, the asynchronous input set-up time is given as parameter #47. If this set-up is met on an input, such as $\overline{DTACK}$, the processor is guaranteed to recognize that signal on the next falling edge of the system clock. However, the converse is not true — if the input signal does not meet the set-up time it is not guaranteed not to be recognized. In addition, if $\overline{DTACK}$ is recognized on a falling edge, valid data will be latched into the processor (on a read cycle) on the next falling edge provided that the data meets the set-up time given as parameter #27. Given this, parameter #31 may be ignored. Note that if $\overline{DTACK}$ is asserted, with the required set-up time, before the falling edge of S4, no wait states will be incurred and the bus cycle will run at its maximum speed of four clock periods.

In order to assure proper operation in a synchronous system when $\overline{BERR}$ is asserted after $\overline{DTACK}$, $\overline{BERR}$ must meet the set-up time parameter #27A prior to the falling edge of the clock one clock cycle after $\overline{DTACK}$ was recognized. This set-up time is critical to proper operation, and the SCN68010 may exhibit erratic behavior if it is violated.

NOTE

During an active bus cycle, $\overline{VPA}$ and $\overline{BERR}$ are sampled on every falling edge of the clock starting with S0. $\overline{DTACK}$ is sampled on every falling edge of the clock starting with S4 and data is latched on the falling edge of S6 during a read. The bus cycle will then be terminated in S7 except when $\overline{BERR}$ is asserted in the absence of $\overline{DTACK}$, in which case it will terminate one clock cycle later in S9.

PROCESSING STATES

This section describes the actions of the SCN68010 which are outside the normal processing associated with the execution of instructions. The functions of the bits in the supervisor portion of the status register are covered: the supervisor/user bit, the trace enable bit, and the processor interrupt priority mask. Finally, the sequence of memory references and actions taken by the processor on exception conditions are detailed.

The SCN68010 is always in one of three processing states: normal, exception, or halted. The normal processing state is that associated with instruction execution; the memory references are to fetch instructions and operands, and to store results. Two special cases of the normal state are the stopped state, which the processor enters when a STOP instruction is executed, and the loop mode, which the processor may enter when a DBcc instruction is executed. In the stopped state, no further memory references are made and in the loop mode only operand references are made.

The exception processing state is associated with interrupts, trap instructions, tracing and other exceptional conditions. The exception may be internally generated by an instruction or by an unusual condition arising during the execution of an instruction. Externally, exception processing can be forced by an interrupt, by a bus error, or by a reset. Exception processing is designed to provide an efficient context switch so that the processor may handle unusual conditions.

The halted processing state is an indication of catastrophic hardware failure. For example, if during the exception processing of a bus error another bus error occurs, the processor assumes that the system is unusable and halts. Only an external reset can restart a halted processor. Note that a processor in the stopped state is not in the halted state, nor vice versa.

Privilege States

The processor operates in one of two states of privilege: the "supervisor" state or the "user" state. The privilege state determines which operations are legal, are used to choose between the supervisor stack pointer and the user stack pointer in instruction references, and may be used by an external memory management device to control and translate accesses.

The privilege state is a mechanism for providing security in a computer system. Programs should access only their own code and data areas, and ought to be restricted from accessing information which they do not need and must not modify.

The privilege mechanism provides security by allowing most programs to execute in user state. In this state, the accesses are controlled, and the effects on other parts of the system are limited. The operating system executes in the supervisor state, has access to all resources, and performs the overhead tasks for the user programs.

Supervisor State

The supervisor state is the higher state of privilege. For instruction execution, the supervisor state is determined by the S bit of the status register; if the S bit is asserted (high), the processor is in the supervisor state. All instructions can be executed in the supervisor state. The bus cycles generated by instructions executed in the supervisor state are classified as supervisor references. While the processor is in the supervisor privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly access the supervisor stack pointer.

All exception processing is done in the supervisor state, regardless of the previous setting of the S bit. The bus cycles generated during exception processing are classified as supervisor references. All stacking operations during exception processing use the supervisor stack pointer.

User State

The user state is the lower state of privilege. For instruction execution, the user state is determined by the S bit of the status register; if the S bit is negated (low), the processor is executing instructions in the user state.

Most instructions execute the same in user state as in the supervisor state. However, some instructions which have important system effects are made privileged. User programs are not permitted to execute the STOP

instruction, or the RESET instruction. To ensure that a user program cannot enter the supervisor state except in a controlled manner, the instructions which modify the whole status register are privileged. To aid in debugging programs which are to be used as operating systems, the move from status register (MOVE from SR), move to/from user stack pointer (MOVE USP), move to/from control register (MOVEC), and move alternate address space (MOVES) instructions are also privileged.

The bus cycles generated by an instruction executed in the user state are classified as user state references. This allows an external memory management device to translate the address and to control access to protected portions of the address space. While the processor is in the user privilege state, those instructions which use either the system stack pointer implicitly or address register seven explicitly, access the user stack pointer.

Privilege State Changes

Once the processor is in the user state and executing instructions, only exception processing can change the privilege state. During exception processing, the previous setting of the S bit of the status register is saved and the S bit is asserted, putting the processor in the supervisor state. Therefore, when instruction execution resumes at the address specified to process the exception, the processor is in the supervisor privilege state.

Reference Classification

When the processor makes a reference, it classifies the kind of reference being made, using the encoding on the three function code output lines. This allows external translation of addresses, control of access, and differentiation of special processor state, such as interrupt acknowledge. Table 18 lists the classification of references.

Table 18. BUS CYCLE CLASSIFICATION

FUNCTION CODE OUTPUT			REFERENCE CLASS
FC2	FC1	FC0	
0	0	0	(Unassigned)
0	0	1	User data
0	1	0	User program
0	1	1	(Unassigned)
1	0	0	(Unassigned)
1	0	1	Supervisor data
1	1	0	Supervisor program
1	1	1	CPU space

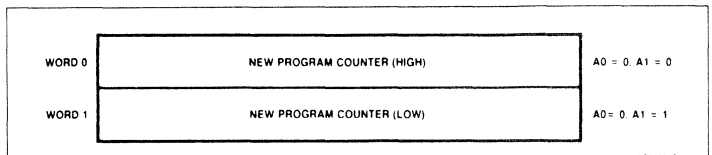


Figure 31. Format of Vector Table Entries

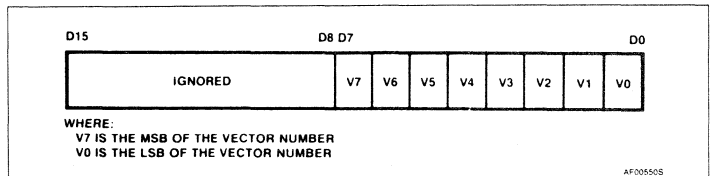


Figure 32. Vector Number Format

Exception Processing

Before discussing the details of interrupts, traps, and tracing, a general description of exception processing is in order. The processing of an exception occurs in four steps, with variations for different exception causes. During the first step, a temporary copy of the status register is made and the status register is set for exception processing. In the second step the exception vector is determined and the third step is the saving of the current processor context. In the fourth step a new context is obtained and the processor resumes instruction processing.

Exception Vectors

Exception vectors are memory locations from which the processor fetches the address of a routine which will handle that exception. All exception vectors are two words in length (figure 31), except for the reset vector, which is four words. All exception vectors lie in the supervisor data space, except for the reset vector which is in the supervisor program space. A vector number is an 8-bit number which, when multiplied by four, gives the offset of an exception vector. Vector numbers are generated internally or externally, depending on the cause of the exception. In the case of interrupts, during the interrupt acknowledge bus cycle, a peripheral provides an 8-bit vector number (figure 32) to the processor on data bus lines D0 through D7. The processor translates the vector number into a full 32-bit offset which is added to the contents of the vector base register to generate the address used to fetch the vector, as shown in figure 33. The memory layout for exception vectors is given in table 19.

Preliminary

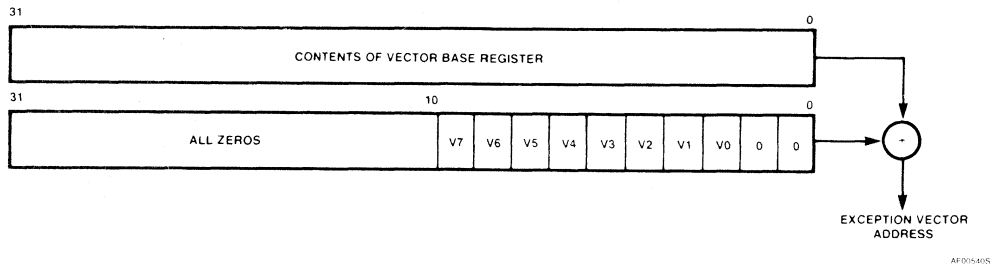


Figure 33. Exception Vector Address Calculation

As shown in table 19, the memory layout is 512 words long (1024 bytes). It starts at offset 0 and proceeds through offset 1023. This provides 255 unique vectors; some of these are reserved for TRAPs and other system functions. Of the 255, there are 192 reserved for user interrupt vectors. However, there is no protection on the first 63 entries, so externally generated interrupt vector numbers may reference any of the exception vectors at the discretion of the system designer.

Exception Stack Frame

Exception processing saves the most volatile portion of the current processor context on the top of the supervisor stack. This context is organized in a format called the exception stack frame. This information always includes the status register and program counter of the processor when the exception occurred. In order to support generic handlers, the processor also places the vector offset in the exception stack frame. The format code field allows the RTE (return from exception) instruction to identify what information is on the stack so that it may be properly restored. The general form of the exception stack frame is illustrated in figure 34. Table 20 lists the SCN68010 stack frame codes. Although some formats are peculiar to a particular S68000 family processor, the format 0000 is always legal, and indicates that just the first four words of the frame are present.

Table 20. SCN68010 FORMAT CODES

FORMAT CODE	STACKED INFORMATION
0000	SCN68010 short format (4 words)
1000	SCN68010 long format (29 words)
All others	Unassigned, reserved

Table 19. EXCEPTION VECTOR TABLE

VECTOR NUMBER(S)	OFFSET			ASSIGNMENT
	DEC	HEX	SPACE	
0	0	000	SP	Reset: initial SSP
—	4	004	SP	Reset: initial PC
2	8	008	SD	Bus error
3	12	00C	SD	Address error
4	16	010	SD	Illegal instruction
5	20	014	SD	Zero divide
6	24	018	SD	CHK instruction
7	28	01C	SD	TRAPV instruction
8	32	020	SD	Privilege violation
9	36	024	SD	Trace
10	40	028	SD	Line 1010 emulator
11	44	02C	SD	Line 1111 emulator
12*	48	030	SD	(Unassigned, reserved)
13*	52	034	SD	(Unassigned, reserved)
14	56	038	SD	Format error
15	60	03C	SD	Uninitialized interrupt vector
16 – 23*	64	04C	SD	(Unassigned, reserved)
	95	05F		—
24	96	060	SD	Spurious interrupt
25	100	064	SD	Level 1 interrupt autovector
26	104	068	SD	Level 2 interrupt autovector
27	108	06C	SD	Level 3 interrupt autovector
28	112	070	SD	Level 4 interrupt autovector
29	116	074	SD	Level 5 interrupt autovector
30	120	078	SD	Level 6 interrupt autovector
31	124	07C	SD	Level 7 interrupt autovector
32 – 47	128	080	SD	TRAP instruction vectors
	191	0BF		—
48 – 63*	192	0C0	SD	(Unassigned, reserved)
	255	0FF		—
64 – 255	256	100	SD	User interrupt vectors

*Vector numbers 12, 13, 16 through 23, and 48 through 63 are reserved for future enhancements by Signetics. No user peripheral devices should be assigned these numbers.

Preliminary

Kinds of Exceptions

Exceptions can be generated by either internal or external causes. The externally generated exceptions are the interrupts, bus error, and reset requests. The interrupts are requests from peripheral devices for processor action while the bus error and reset inputs are used for access control and processor restart. The internally generated exceptions come from instructions, or from address errors or tracing. The trap (TRAP), trap on overflow (TRAPV), check data register against upper bounds (CHK), and divide (DIV) instructions all can generate exceptions as part of their instruction execution. In addition, illegal instructions, word or long word fetches from odd addresses, and privilege violations cause exceptions. Tracing behaves like a very high-priority internally-generated interrupt after each instruction execution.

Exception Processing Sequence

Exception processing occurs in four identifiable steps. In the first step, an internal copy is made of the status register. After the copy is made, the S bit is asserted, putting the processor into the supervisor privilege state. Also, the T bit is negated which will allow the exception handler to execute unhindered by tracing. For the reset and interrupt exceptions, the interrupt priority mask is also updated.

In the second step, the vector number of the exception is determined. For interrupts, the vector number is obtained by a processor fetch classified as an interrupt acknowledge. For all other exceptions, internal logic provides the vector number. This vector number is then used to generate the offset of the exception vector and is added to the vector base register.

The third step is to save the current processor status, except for the reset exception. The exception stack frame is created at the top of the supervisor stack. The current program counter value, the saved copy of the status register, and the format/offset word are written into the stack frame. The program counter value stacked usually points to the next unexecuted instruction; however, for bus error and address error, the value stacked for the program counter is unpredictable, and may be incremented by up to five words from the address of the instruction which caused the error. Group 1 and 2 exceptions (see **Multiple Exceptions**) use a short format exception stack frame (format = 0000). Additional information defining the current context is stacked for the bus error and address error exceptions.

The last step is the same for all exceptions. The new program counter value is fetched from the exception vector table. The processor then resumes instruction execution. The instruction at the address given in the exception vector is fetched, and normal instruction decoding and execution is started.

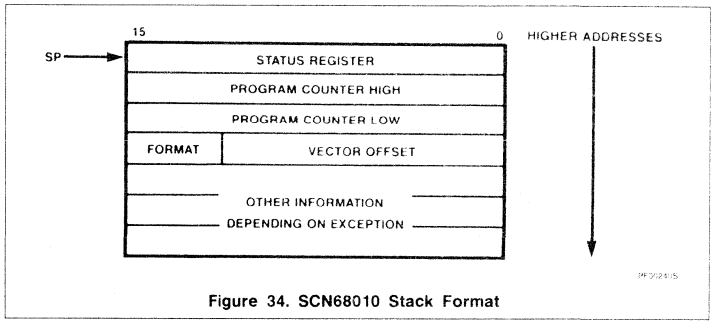


Figure 34. SCN68010 Stack Format

Multiple Exceptions

These paragraphs describe the processing which occurs when multiple exceptions arise simultaneously. Exceptions can be grouped according to their occurrence and priority. The group 0 exceptions are reset, bus error, and address error. These exceptions cause the instruction currently being executed to be aborted and the exception processing to commence within two clock cycles. The group 1 exceptions are trace and interrupt, as well as the privilege violations and illegal instructions. These exceptions allow the current instruction to execute to completion, but pre-empt the execution of the next instruction by forcing exception processing to occur (privilege violations and illegal instructions are detected when they are the next instruction to be executed). The group 2 exceptions occur as part of the normal processing of instructions. The TRAP, TRAPV, CHK, and zero divide exceptions are in this group. For these exceptions, the normal execution of an instruction may lead to exception processing.

Group 0 exceptions have highest priority, while group 2 exceptions have lowest priority. Within group 0, reset has highest priority, followed by address error and then bus error. Within group 1, trace has priority over external interrupts, which in turn takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, there is no priority relation within group 2.

The priority relation between two exceptions determines which is taken, or taken first, if the conditions for both arise simultaneously. Therefore, if a bus error occurs during a TRAP instruction, the bus error takes precedence, and the TRAP instruction processing is suspended. In another example, if an interrupt request occurs during the execution of an instruction while the T bit is asserted, the trace exception has priority, and is processed first. Before instruction processing resumes, however, the interrupt exception is also processed, and instruction processing commences finally in the interrupt handler routine. A summary of exception grouping and priority is given in table 21.

Exception Processing in Detail

Exceptions have a number of sources and each exception has processing which is peculiar to it. The following paragraphs detail the sources of exceptions, how each arises, and how each is processed.

Reset

The reset input provides the highest exception level. The processing of the reset signal is designed for system initiation, and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The processor is forced into the supervisor state, the trace state is forced off, and the processor interrupt priority mask is set to level seven. The vector base register is set to $00000000 and the vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents , in particular the supervisor stack pointer, neither the program counter nor the status register is saved. The address contained in the first two words of the reset exception vector is fetched as the initial supervisor stack pointer, and the address in the last two words of the reset exception vector is fetched as the initial program counter. Finally, instruction execution is started at the address in the program counter. The power-up/restart code should be pointed to by the initial program counter.

The reset instruction does not cause loading of the reset vector, but does assert the reset line to reset external devices. This allows the software to reset the system to a known state and then continue processing with the next instruction.

Preliminary

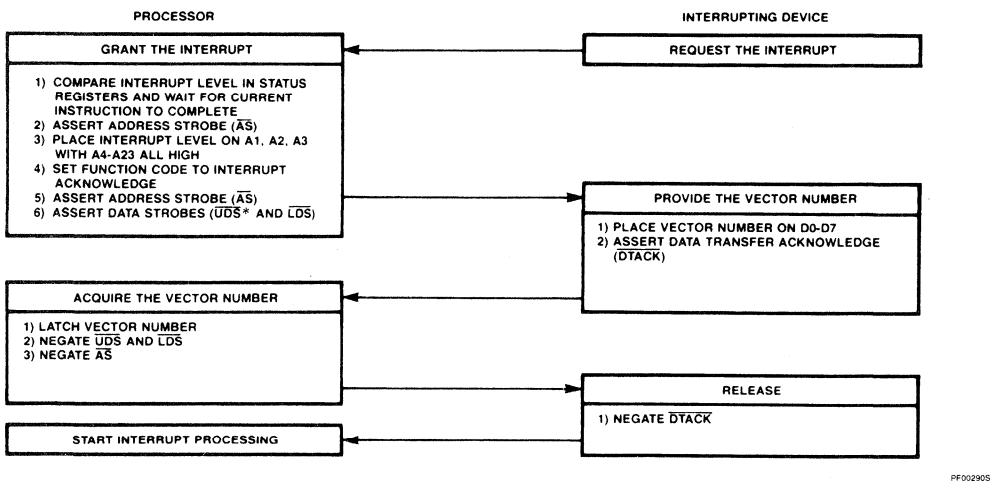


Figure 35. Vector Acquisition Flowchart

Interrupts

Seven levels of interrupt priorities are provided. Devices may be chained externally within interrupt priority levels, allowing an unlimited number of peripheral devices to interrupt the processor. Interrupt priority levels are numbered from one to seven, level seven being the highest priority. The status register contains a 3-bit mask which indicates the current processor priority, and interrupts are inhibited for all priority levels less than or equal to the current processor priority.

An interrupt request is made to the processor by encoding the interrupt request level on the interrupt request lines; a zero indicates no interrupt request. Interrupt requests arriving at the processor do not force immediate exception processing, but are made pending. Pending interrupts may cause exception processing to start at the end of an instruction depending on the current processor priority level. If the priority of the pending interrupt is lower than or equal to the current processor priority, execution continues with the next instruction and the interrupt exception processing is postponed. (The recognition of level seven is slightly different, as explained in the following paragraph.)

If the priority of the pending interrupt is greater than the current processor priority, the exception processing sequence is started. A copy of the status register is saved, the privilege state is set to supervisor state, tracing is suppressed, and the processor priority level is set to the level of the interrupt being acknowledged. The processor fetches the vector number from the interrupting device, classifying the reference as an interrupt acknowledge and displaying the level number of the interrupt being acknowledged on the address bus. If external logic requests automatic vectoring, the processor internally generates a vector number which is determined by the interrupt level number. If external logic indicates a bus error, the interrupt is taken to be spurious, and the generated vector number references the spurious interrupt vector. The processor then proceeds with the usual exception processing, saving the format/offset word, program counter, and status register on the supervisor stack. The offset value in the format/offset word is the externally supplied or internally generated vector number multiplied by four. The format will be all zeros. The saved value of the program counter is the address of the instruction which would have been executed had the interrupt not been present. The content of the interrupt vector whose vector number was previously obtained is fetched and loaded into the program counter, and normal instruction execution commences in the interrupt handling routine. A flowchart for the interrupt acknowledge sequence is given in figure 35, a timing diagram is given in figure 36, and the interrupt processing sequence is shown in figure 37.

Table 21. EXCEPTION GROUPING AND PRIORITY

GROUP	EXCEPTION	PROCESSING
0	Reset address error bus error	Exception processing begins within two clock cycles
1	Trace interrupt illegal privilege	Exception processing begins before the next instruction
2	TRAP, TRAPV, CHK, zero divide format error	Exception processing is started by normal instruction execution

Preliminary

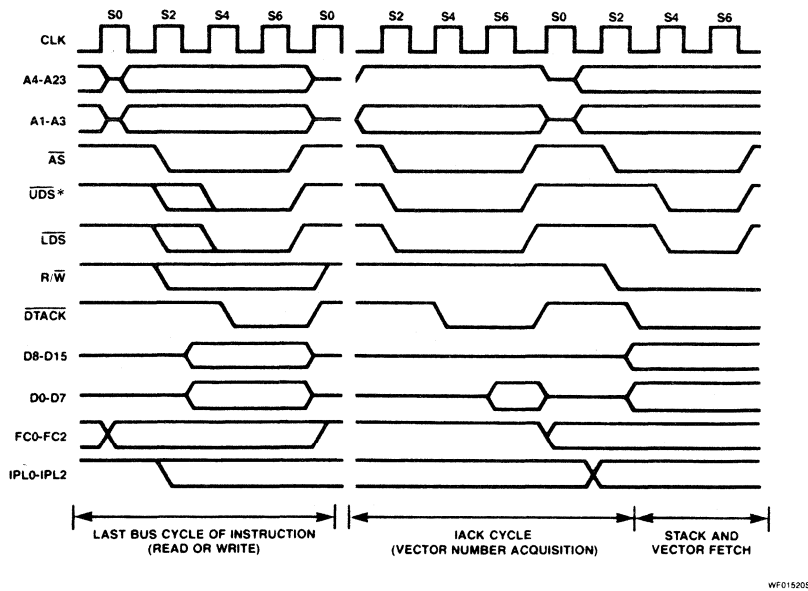


Figure 36. Interrupt Acknowledge Cycle Timing Diagram

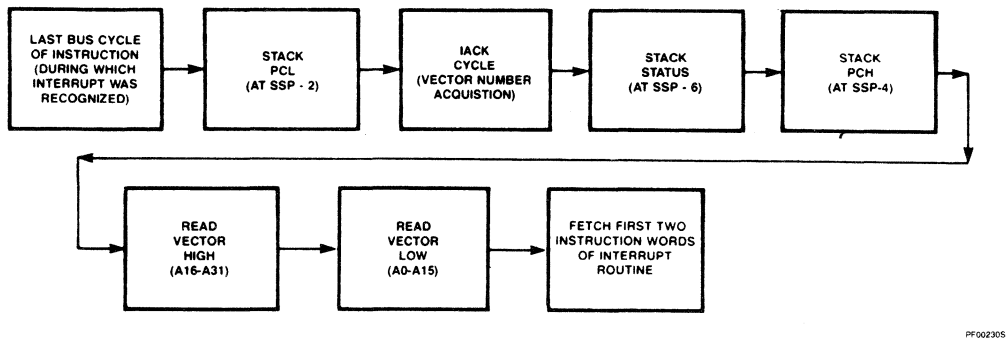


Figure 37. Interrupt Processing Sequence

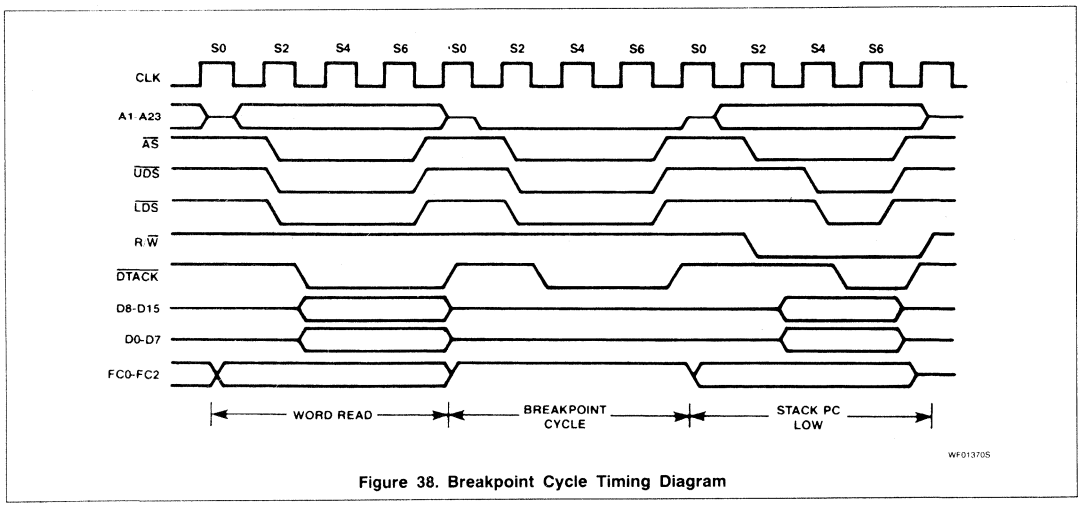


Figure 38. Breakpoint Cycle Timing Diagram

Priority level seven is a special case. Level seven interrupts cannot be inhibited by the interrupt priority mask, thus providing a "non-maskable interrupt" capability. An interrupt is generated each time the interrupt request level changes from some lower level to level seven. Note that a level seven interrupt may still be caused by the level comparison if the request level is a seven and the processor priority is set to a lower level by an instruction.

Uninitialized Interrupt

An interrupting device asserts $\overline{VPA}$, $\overline{BERR}$, or provides an interrupt vector number and asserts $\overline{DTACK}$ during an interrupt acknowledge cycle by the SCN68010. If the vector register has not been initialized, the responding S68000 family peripheral will provide vector number 15, the uninitialized interrupt vector. This provides a uniform way to recover from a programming error.

Spurious Interrupt

If during the interrupt acknowledge cycle no device responds by asserting $\overline{DTACK}$ or $\overline{VPA}$, $\overline{BERR}$ should be asserted to terminate the vector acquisition. The processor separates the processing of this error from bus error by forming a short format exception stack and fetching the spurious interrupt vector instead of the bus error vector. The processor then proceeds with the usual exception processing.

Instruction Traps

Traps are exceptions caused by instructions. They arise either from processor recognition of abnormal conditions during instruction execution, or from use of instructions whose normal behavior is trapping.

Some instructions are used specifically to generate traps. The TRAP instruction always forces an exception and is useful for implementing supervisor calls for user programs. The TRAPV and CHK instructions force an exception if the user program detects a runtime error, which may be an arithmetic overflow or a subscript out of bounds.

The signed divide (DIVS) and unsigned divide (DIVU) instructions will force an exception if a division operation is attempted with a divisor of zero.

Illegal and Unimplemented Instructions

Illegal instruction is the term used to refer to any of the word bit patterns which are not the bit patterns of the first word of a legal SCN68010 instruction. During instruction execution, if such an instruction is fetched, an illegal instruction exception occurs. Signetics reserves the right to define instructions whose opcodes may be any of the illegal instructions. Three bit patterns will always force an illegal instruction trap on all S68000 family compatible microprocessors. They are: $4AFA, $4AFB, and $4AFC. Two of the patterns, $4AFA and $4AFB, are reserved for Signetics system products. The third pattern, $4AFC, is reserved for customer use.

In addition to the previously defined illegal instruction opcodes, the SCN68010 defines eight breakpoint illegal instructions with the bit patterns $4848 – $484F. These instructions cause the processor to enter illegal instruction exception processing as usual, but a breakpoint bus cycle is executed before the stacking operations are performed as shown in figure 38. The processor does not accept or send any data during this cycle. Whether the breakpoint cycle is terminated with a $\overline{DTACK}$, $\overline{BERR}$, or $\overline{VPA}$ signal, the processor will continue with the illegal instruction processing. The purpose of this cycle is to provide a software breakpoint that will signal external hardware when it is executed.

Preliminary

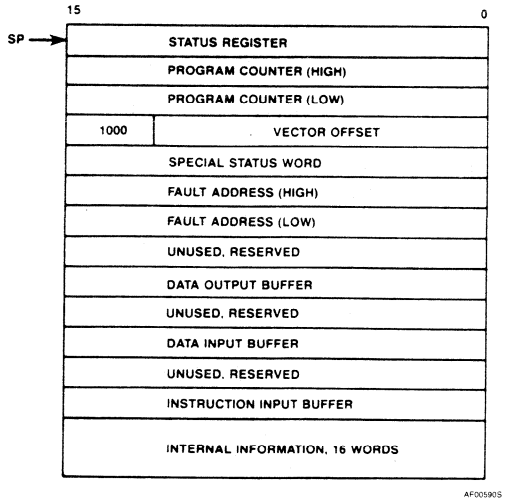


Figure 39. Exception Stack Order (Bus and Address Error)

Word patterns with bits 12 – 15 equaling 1010 or 1111 are distinguished as unimplemented instructions and separate exception vectors are given to these patterns to permit efficient emulation. This facility allows the operating system to detect program errors, or to emulate unimplemented instructions in software.

Privilege Violations

In order to provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user state will cause an exception. The privileged instructions are:

AND immediate to SR	MOVE USP
EOR immediate to SR	OR immediate to SR
MOVE to SR	RESET
MOVE from SR	RTE
MOVEC	STOP
MOVES	

Tracing

To aid in program development, the SCN68010 includes a facility to allow instruction-by-instruction tracing. In the trace state, after each instruction is executed an exception is forced, allowing a debugging program to monitor the execution of the program under test.

The trace facility uses the T bit in the supervisor portion of the status register. If the T bit is negated (off), tracing is disabled, and instruction execution proceeds from instruction to instruction as normal. If the T bit is asserted (on) at the beginning of the execution of an instruction, a trace exception will be generated as the execution of that instruction is completed. If the instruction is not executed, either because an interrupt is taken, or the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is indeed executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. If, during the execution of the instruction an exception is forced by that instruction, the forced exception is processed before the trace exception.

As an extreme illustration of the above rules, consider the arrival of an interrupt during the execution of a TRAP instruction while tracing is enabled. First the trap exception is processed, then the trace exception, and finally the interrupt exception. Instruction execution resumes in the interrupt handler routine.

Bus Error

Bus error exceptions occur when external logic terminates a bus cycle with a bus error signal. Whether the processor was doing instruction or exception processing, that processing is terminated, and the processor immediately begins exception processing. However, if a bus error occurs during exception processing for a bus error, address error, or reset, the processor detects a double bus fault and halts. When exception processing is completed, instruction execution continues at the address contained in exception vector table entry two, at offset $008.

Exception processing for a bus error follows a slightly different sequence than the sequence for group 1 and 2 exceptions. In addition to the four steps executed during exception processing for all other exceptions, 22 words of additional information are placed on the stack. This additional information describes the internal state of the processor at the time of the bus error and is reloaded by the RTE instruction to continue the instruction that caused the error. Figure 39 shows the order of the stacked information.

The value of the saved program counter does not necessarily point to the instruction that was executing when the bus error occurred, but may be advanced by up to five words. This is due to the prefetch mechanism on the SCN68010 that always fetches a new instruction word as each previously fetched instruction word is used (see **Instruction Prefetch**). However, enough information is placed on the stack for the bus error exception handler routine to determine why the bus fault occurred. This additional information includes the address that was being accessed, the function codes for the access, whether it was a read or a write, and what internal register was included in the transfer. The fault address can be used by an operating system to determine what virtual memory location is needed so that the requested data can be brought into physical memory. The RTE instruction is then used to reload the processor's internal state at the time of the fault, the faulted bus cycle will then be re-run and the suspended instruction completed. If the faulted bus cycle was a read-modify-write, the entire cycle will be re-run whether the fault occurred during the read or the write operation.

Preliminary

An alternate method of handling a bus error is to complete the faulted access in software. In order to use this method, use of the special status word, the instruction input buffer, the data input buffer, and the data output buffer image is required. The format of the special status word is shown in figure 40. If the bus cycle was a write, the data output buffer image should be written to the fault address location using the function code contained in the special status word. If the cycle was a read, the data at the fault address location should be written to the images of the data input buffer, instruction input buffer, or both according to the DF and IF bits.[1] In addition, for read-modify-write cycles, the status register image must be properly set to reflect the read data if the fault occurred during the read portion of the cycle and the write operation (i.e., setting the most significant bit of the memory location) must also be performed. This is because the entire read-modify-write cycle is assumed to have been completed by software. Once the cycle has been completed by software, the RR bit in the special status word is set to indicate to the processor that it should not re-run the cycle when the RTE instruction is executed. If the re-run flag is set when an RTE instruction executes, the SCN68010 still reads all of the information from the stack.

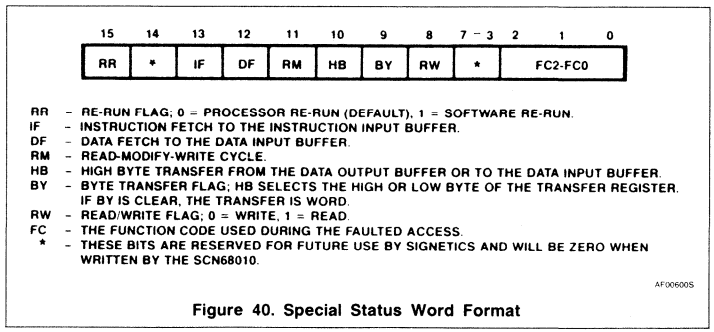


Figure 40. Special Status Word Format

Address Error

Address error exceptions occur when the processor attempts to access a word or long word operand or an instruction at an odd address. The effect is much like an internally generated bus error, so that the bus cycle is aborted, and the processor begins exception processing. After exception processing commences, the sequence is the same as that for bus error including the information that is stacked, except that the vector offset refers to the address error exception vector. If an address error occurs during exception processing for a bus error, address error, or reset, the processor detects a double bus fault and halts.

As shown in figure 41, an address error will execute a short bus cycle followed by exception processing. This short bus cycle is similar to a normal read or write cycle, except that the data strobes are not asserted and no external signals are used to terminate the cycle. During an address error bus cycle, AS is asserted to indicate that the SCN68010 will drive the address bus (thus allowing for proper operation in a multiple bus master system). Note that data strobes are not asserted allowing for address error detection and memory protection.

Since the address error exception stacks the same information that is stacked by a bus error exception, it is possible to use the RTE instruction to continue execution of the suspended instruction. However, if the software re-run flag is not set, the fault address will be used when the cycle is re-run and another address error exception will occur. Therefore, the user must be certain that the proper corrections have been made to the stack image and user registers before attempting to continue the instruction. With proper software handling, the address error exception handler could emulate word or long word accesses to odd addresses if desired.

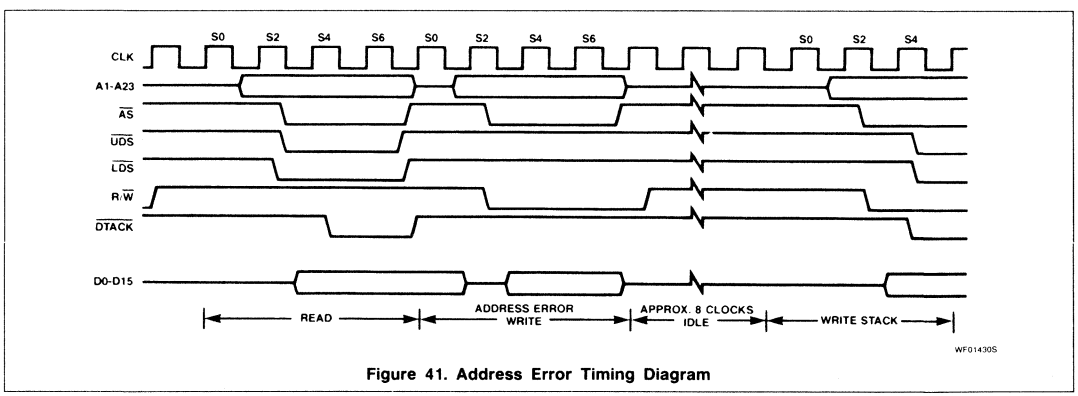


Figure 41. Address Error Timing Diagram

[1] If the faulted access was a byte operation, the data should be moved from or to the least-significant byte of the data output or input buffer images unless the HB bit is set. This condition will only occur if a MOVEP instruction caused the fault during the transfer of bits 8 – 15 of a word or long word or bits 24 – 31 of a long word.

Preliminary

Return from Exception

In addition to returning from any exception handler routine, the RTE instruction is used to resume the execution of a suspended instruction by restoring all of the temporary register and control information stored during a bus error and returning to the normal processing state. For the RTE instruction to execute properly, the stack must contain valid and accessible data. The RTE instruction checks for data validity in two ways; first, by checking the format/offset word for a valid stack format code, and second, if the format code indicates the long stack format, the long stack data is checked for validity as it is loaded into the processor. In addition, the data is checked for accessibility when the processor starts reading the long data. Because of these checks, the RTE instruction executes as follows:

1. Determine the stack format. This step is the same for any stack format and consists of reading the status register, program counter, and format/offset word. If the format code indicates a short stack format, execution continues at the new program counter address. If the format code is not one of the SCN68010 defined stack format codes, exception processing starts for a format error.
2. Determine data validity. For a long stack format, the SCN68010 will begin to read the remaining stack data, checking for validity of the data. The only word checked for validity is the first of the 16 internal information words (SP + 26) shown in figure 39. This word contains a processor version number in addition to proprietary internal information that must match the version number of the SCN68010 that is attempting to read the data. This validity check is used to insure that in dual processor systems, the data will be properly interpreted by the RTE instruction if the two processors are of different versions. If the version number is incorrect for this processor, the RTE instruction will be aborted and exception processing will begin for a format error exception. Since the stack pointer is not updated until the RTE instruction has successfully read all of the stack data, a format error occurring at this point will not stack new data over the previous bus error stack information.
3. Determine data accessibility. If the long stack data is valid, the SCN68010 performs a read from the last word (SP + 56) of the long stack to determine data accessibility. If this read is terminated normally, the processor assumes that the remaining words on the stack frame are also accessible. If a bus error is signaled before or during this read, a bus error exception is taken as usual. After this read, the processor must be able to load the remaining data without receiving a bus error; therefore, if a bus error occurs on any of the remaining stack reads, the SCN68010 treats this as a double bus fault and enters the halted state.

INTERFACE WITH SYNCHRONOUS PERIPHERALS

To interface the synchronous peripherals with the asynchronous SCN68010, the processor modifies its bus cycle to meet the synchronous cycle requirements whenever a synchronous device address is detected. This is possible since both processors use memory mapped I/O. Figure 42 is a flowchart of the interface operation between the processor and synchronous devices.

Data Transfer Operation

Three signals on the processor provide the synchronous interface. They are: enable (E), valid memory address ($\overline{VMA}$), and valid peripheral address ($\overline{VPA}$). Enable corresponds to the E or phase 2 signal in existing synchronous systems. The bus frequency is one tenth of the incoming SCN68010 clock frequency. The timing of E allows 1 megahertz peripherals to be used with an 8 megahertz SCN68010. Enable has a 60/40 duty cycle; that is, it is low for six input clocks and high for four input clocks. This duty cycle allows the processor to do successive $\overline{VPA}$ accesses on successive E pulses.

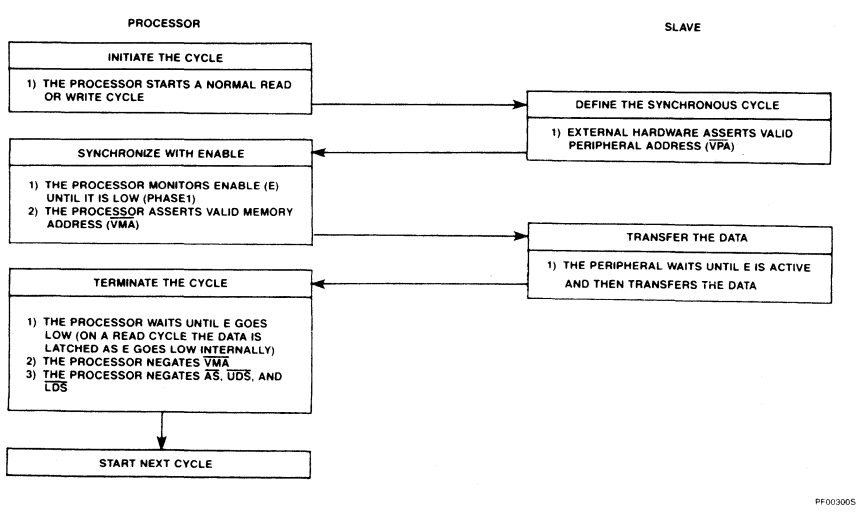


Figure 42. Synchronous Interfacing Flowchart

Preliminary

Synchronous cycle timing is given in figure 43. At state zero (S0) in the cycle, the address bus is in the high-impedance state. A function code is asserted on the function code output lines. One-half clock later, in state 1, the address bus is released from the high-impedance state.

During state 2, the address strobe ($\overline{AS}$) is asserted to indicate that there is a valid address on the address bus. If the bus cycle is a read cycle, the upper and/or lower data strobes are also asserted in state 2. If the bus cycle is a write cycle, the read/write (R/$\overline{W}$) signal is switched to low (write) during state 2. One-half clock later, in state 3, the write data is placed on the data bus, and in state 4 the data strobes are issued to indicate valid data on the data bus. The processor now inserts wait states until it recognizes the assertion of $\overline{VPA}$.

The $\overline{VPA}$ input signals the processor that the address on the bus is the address of a synchronous device (or an area reserved for synchronous devices) and that the bus should conform to the phase 2 transfer characteristics of the synchronous bus. Valid peripheral address is derived by decoding the address bus, conditioned by the address strobe. Chip select for the synchronous peripherals should be derived by decoding the address bus conditioned by $\overline{VMA}$.

After recognition of $\overline{VPA}$, the processor assures that the enable (E) is low, by waiting if necessary, and subsequently asserts $\overline{VMA}$ two clock cycles before E goes high. $\overline{VMA}$ is then used as part of the chip select equation of the peripheral. This ensures that the synchronous peripherals are selected and deselected at the correct time. The peripheral now runs its cycle during the high portion of the E signal. Figures 43 and 44 depict the best and worst case synchronous cycle timing; this cycle length is dependent strictly upon when $\overline{VPA}$ is asserted in relationship to the E clock.

If we assume that external circuitry asserts $\overline{VPA}$ as soon as possible after the assertion of $\overline{AS}$, then $\overline{VPA}$ will be recognized as being asserted on the falling edge of S4. In this case, no "extra" wait cycles will be inserted prior to the recognition of $\overline{VPA}$ asserted and only the wait cycles inserted to synchronize with the E clock will determine the total length of the cycle. In any case, the synchronization delay will be some integral number of clock cycles within the following two extremes:

1. Best Case — $\overline{VPA}$ is recognized as being asserted on the falling edge three clock cycles before E rises (or three clock cycles after E falls).
2. Worst Case — $\overline{VPA}$ is recognized as being asserted on the falling edge two clock cycles before E rises (or four clock cycles after E falls).

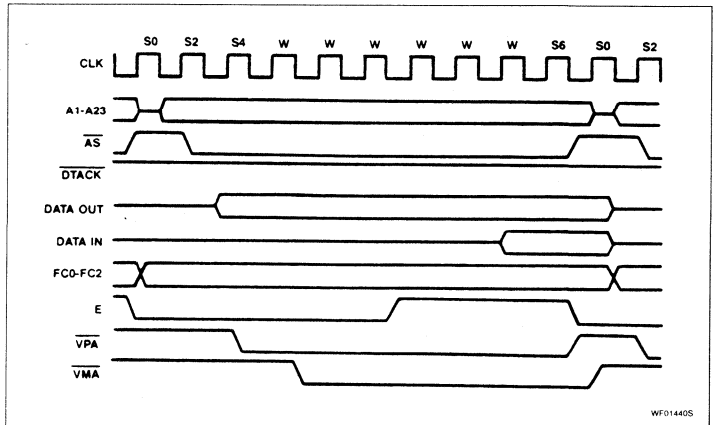


Figure 43. SCN68010 to Synchronous Peripheral Timing Diagram — Best Case

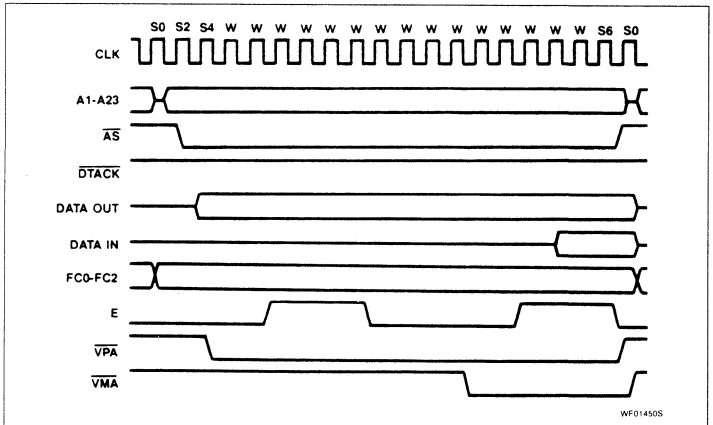


Figure 44. SCN68010 to Synchronous Peripheral Timing Diagram — Worst Case

During a read cycle, the processor latches the peripheral data in state 6. For all cycles, the processor negates the address and data strobes one-half clock cycle later in state 7 and the enable signal goes low at this time. Another half clock later, the address bus is put in the high-impedance state. During a write cycle, the data bus is put in the high-impedance state and the read/write signal is switched high. The peripheral logic must remove $\overline{VPA}$ within one clock after the address strobe is negated.

$\overline{DTACK}$ should not be asserted while $\overline{VPA}$ is asserted. Notice that the SCN68010 $\overline{VMA}$ is active low, contrasted with the active high synchronous VMA. This allows the processor to put its buses in the high-impedance state on DMA requests without inadvertently selecting the peripherals.

Preliminary

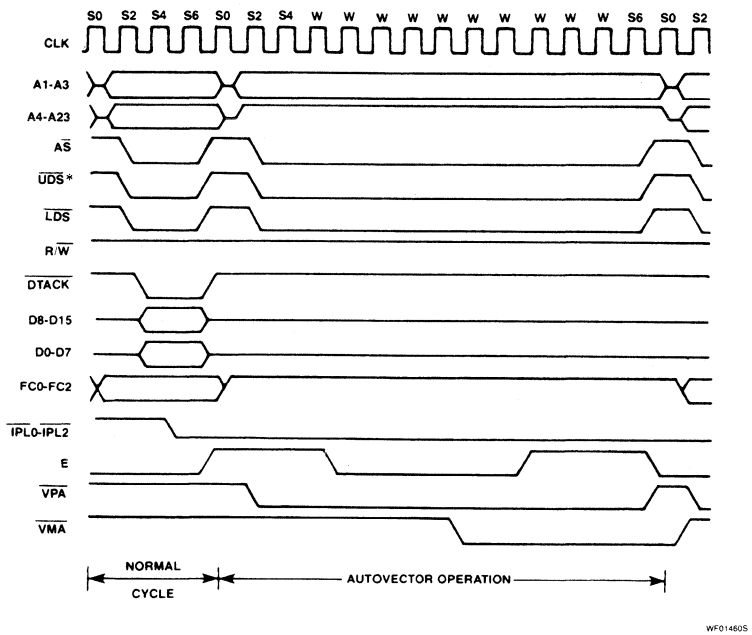


Figure 45. Autovector Operation Timing Diagram

Interrupt Interface Operation

During an interrupt acknowledge cycle while the processor is fetching the vector, if $\overline{VPA}$ is asserted the SCN68010 will assert $\overline{VMA}$ and complete a normal synchronous read cycle as shown in figure 45. The processor will then use an internally generated vector that is a function of the interrupt being serviced. This process is known as autovectoring. The seven autovectors are vector numbers 25 through 31 (decimal).

Autovectoring operates in the same fashion (but is not restricted to) the synchronous interrupt sequence. The basic difference is that there are six normal interrupt vectors and one NMI type vector. As with both the synchronous and the SCN68010's normal vectored interrupt, the interrupt service routine can be located anywhere in the address space. This is due to the fact that while the vector numbers are fixed, the contents of the vector table entries are assigned by the user.

Since $\overline{VMA}$ is asserted during autovectoring, the synchronous peripheral address decoding should prevent unintended accesses.

Preliminary

INSTRUCTION SET AND EXECUTION TIMES

Instruction Set

The following paragraphs provide information about the addressing categories and instruction set of the SCN68010.

Addressing Categories

Effective address modes may be categorized by the ways in which they may be used. The following classifications will be used in the instruction definitions:

Data — If an effective address mode may be used to refer to data operands, it is considered a data addressing effective address mode.

Memory — If an effective address mode may be used to refer to memory operands, it is considered a memory addressing effective address mode.

Alterable — If an effective address mode may be used to refer to alterable (writeable) operands, it is considered an alterable addressing effective address mode.

Control — If an effective address mode may be used to refer to memory operands without an associated size, it is considered a control addressing effective address mode.

Table 22. EFFECTIVE ADDRESSING MODE CATEGORIES

EFFECTIVE ADDRESS MODES	MODE	REGISTER	DATA	ADDRESSING CATEGORIES		
				Memory	Control	Alterable
Dn	000	Register number	X	–	–	X
An	001	Register number	–	–	–	X
(An)	010	Register number	X	X	X	X
(An)+	011	Register number	X	X	–	X
–(An)	100	Register number	X	X	–	X
d(An)	101	Register number	X	X	X	X
d(An, ix)	110	Register number	X	X	X	X
xxx.W	111	000	X	X	X	X
xxx.L	111	001	X	X	X	X
d(PC)	111	010	X	X	X	–
d(PC, ix)	111	011	X	X	X	–
#xxx	111	100	X	X	–	–

These categories may be combined, so that additional, more restrictive, classifications may be defined. For example, the instruction descriptions use such classifications as alterable memory or data alterable. The former refers to those addressing modes which are both alterable and memory addresses, and the latter refers to addressing modes which are both data and alterable. Table 22 shows the various categories to which each of the effective address modes belong. Table 23 is the instruction set summary.

Preliminary

Table 23. INSTRUCTION SET

MNEMONIC	DESCRIPTION	OPERATION	CONDITION CODES				
			X	N	Z	V	C
ABCD	Add decimal with extend	(Destination)$_{10}$ + (Source)$_{10}$ + X → Destination	*	U	*	U	*
ADD	Add binary	(Destination) + (Source) → Destination	*	*	*	*	*
ADDA	Add address	(Destination) + (Source) → Destination	–	–	–	–	–
ADDI	Add immediate	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDQ	Add quick	(Destination) + Immediate Data → Destination	*	*	*	*	*
ADDX	Add extended	(Destination) + (Source) + X= Destination	*	*	*	*	*
AND	AND logical	(Destination) ∧ Source → Destination	–	*	*	0	0
ANDI	AND immediate	(Destination) ∧ Immediate Data → Destination	–	*	*	0	0
ANDI to CCR	AND immediate to condition codes	(Source) ∧ CCR → CCR	*	*	*	*	*
ANDI to SR	AND immediate to status register	(Source) ∧ SR → SR	*	*	*	*	*
ASL, ASR	Arithmetic shift	(Destination) Shifted by < count > → Destination	*	*	*	*	*
B_{CC}	Branch conditionally	If $_{CC}$ the PC + d → PC	–	–	–	–	–
BCHG	Test a bit and change	~(< bit number >) OF Destination → Z ~(< bit number >) OF Destination → < bit number > OF Destination	–	–	*	–	–
BCLR	Test a bit and clear	~(< bit number >) OF Destination → Z 0 → < bit number > → OF Destination	–	–	*	–	–
BRA	Branch always	(PC) + displacement → PC	–	–	–	–	–
BSET	Test a bit and set	~(< bit number >) OF Destination → Z 1 → < bit number > OF Destination	–	–	*	–	–
BSR	Branch to subroutine	(PC) → –(SP); (PC) + d → PC	–	–	–	–	–
BTST	Test a bit	~(< bit number >) OF Destination → Z	–	–	*	–	–
CHK	Check register against bounds	If Dn < 0 or Dn > (< ea >) then TRAP	–	*	U	U	U
CLR	Clear and operand	0 → Destination	–	0	1	0	0
CMP	Compare	(Destination) – (Source)	–	*	*	*	*
CMPA	Compare address	(Destination) – (Source)	–	*	*	*	*
CMPI	Compare immediate	(Destination) – Immediate Data	–	*	*	*	*
CMPM	Compare memory	(Destination) – (Source)	–	*	*	*	*
DB_{CC}	Test condition, decrement and branch	if ~$_{CC}$ then Dn – 1 → Dn; if Dn ≠ –1 then PC + d → PC	–	–	–	–	–
DIVS	Signed divide	(Destination)/(Source) → Destination	–	*	*	*	0
DIVU	Unsigned divide	(Destination)/(Source) → Destination	–	*	*	*	0
EOR	Exclusive OR logical	(Destination) ⊕ (Source) → Destination	–	*	*	0	0
EORI	Exclusive OR immediate	(Destination) ⊕ Immediate Data → Destination	–	*	*	0	0
EORI to CCR	Exclusive OR immediate to condition codes	(Source) ⊕ CCR → CCR	*	*	*	*	*
EORI to SR	Exclusive OR immediate to status register	(Source) ⊕ SR → SR	*	*	*	*	*
EXG	Exchange register	(Rx) ↔ (Ry)	–	–	–	–	–
EXT	Sign extend	(Destination) Sign-Extended → Destination	–	*	*	0	0
JMP	Jump	(Destination) → PC	–	–	–	–	–
JSR	Jump to subroutine	(PC) → –(SP); Destination → PC	–	–	–	–	–
LEA	Load effective address	Destination → An	–	–	–	–	–
LINK	Link and allocate	(An) → –(SP); (SP) → An; (SP) + → SP	–	–	–	–	–
LSL, LSR	Logical shift	(Destination) Shifted by < count > → Destination	*	*	*	0	*
MOVE	Move data from source to destination	(Source) → Destination	–	*	*	0	0
MOVE to CCR	Move to condition code	(Source) → CCR	*	*	*	*	*
MOVE from CCR	Move from condition codes	(CCR) → Destination	–	–	–	–	–
MOVE to SR	Move to the status register	(Source) → SR	*	*	*	*	*
MOVE from SR	Move from the status register	(SR) → Destination	–	–	–	–	–
MOVE USP	Move user stack pointer	(USP) → An; (An)=Ar USP	–	–	–	–	–
MOVEA	Move address	(Source) → Destination	–	–	–	–	–
MOVEC	Move control register	(Cr) → Rn; (Rn) → Cr	–	–	–	–	–

Preliminary

Table 23. INSTRUCTION SET (Continued)

MNEMONIC	DESCRIPTION	OPERATION	CONDITION CODES				
			X	N	Z	V	C
MOVEM	Move multiple registers	(Registers) → Destination (Source) → Registers	–	–	–	–	–
MOVEP	Move peripheral data	(Source) → Destination	–	–	–	–	–
MOVEQ	Move quick	Immediate Data → Destination	–	*	*	0	0
MOVES	Move alternate address space	(Dn) → Destination; (Source) → Dn	–	–	–	–	–
MULS	Signed multiply	(Destination)X (Source) → Destination	–	*	*	0	0
MULU	Unsigned multiply	(Destination)X (Source) → Destination	–	*	*	0	0
NBCD	Negate decimal with extend	$0 - (\text{Destination})_{10} - X$ → Destination	*	U	*	U	*
NEG	Negate	0 – (Destination) → Destination	*	*	*	*	*
NEGX	Negate with extend	0 – (Destination) – X → Destination	*	*	*	*	*
NOP	No operation	–	–	–	–	–	–
NOT	Logical complement	~ (Destination) → Destination	–	*	*	0	0
OR	Inclusive OR logical	(Destination) v (Source) → Destination	–	*	*	0	0
ORI	Inclusive OR immediate	(Destination) v Immediate Data → Destination	–	*	*	0	0
ORI to CCR	Inclusive OR immediate to condition codes	(Source) v CCR → CCR	*	*	*	*	*
ORI to SR	Inclusive OR immediate to status register	(Source) v SR → SR	*	*	*	*	*
PEA	Push effective address	Destination → – (SP)	–	–	–	–	–
RESET	Reset external device	–	–	–	–	–	–
ROL, ROR	Rotate (without extend)	(Destination) Rotated by < count > → Destination	–	*	*	0	*
ROXL, ROXR	Rotate with extend	(Destination) Rotated by < count > → Destination	*	*	*	0	*
RTD	Return and deallocate stack	(SP)+ → PC; (SP) + d → SP	–	–	–	–	–
RTE	Return from exception	(SP) + → SR; (SP) + → PC	*	*	*	*	*
RTR	Return and restore condition codes	(SP) + → CC; (SP) + → PC	*	*	*	*	*
RTS	Return from subroutine	(SP) + → PC	–	–	–	–	–
SBCD	Subtract decimal with extend	$(\text{Destination})_{10} - (\text{Source})_{10} - X$ → Destination	*	U	*	U	*
S_{CC}	Set according to condition	If $_{CC}$ then 1's → Destination else 0's → Destination	–	–	–	–	–
STOP	Load status register and stop	Immediate Data → SR; STOP	*	*	*	*	*
SUB	Subtract binary	(Destination) – (Source) → Destination	*	*	*	*	*
SUBA	Subtract address	(Destination) – (Source) → Destination	–	–	–	–	–
SUBI	Subtract immediate	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBQ	Subtract quick	(Destination) – Immediate Data → Destination	*	*	*	*	*
SUBX	Subtract with extend	(Destination) – (Source) – X → Destination	*	*	*	*	*
SWAP	Swap register halves	Register [31:16] ↔ Register [15:0]	–	*	*	0	0
TAS	Test and set an operand	(Destination) Tested → CC; 1 → [7] OF Destination	–	*	*	0	0
TRAP	Trap	(PC) → – (SSP); (SR) → – (SSP); (Vector) → PC	–	–	–	–	–
TRAPV	Trap on overflow	If V set then TRAP	–	–	–	–	–
TST	Test and operand	(Destination) Tested → CC	–	*	*	0	0
UNLK	Unlink	(An) → SP; (SP) + → An	–	–	–	–	–

NOTES:

[ ] = bit number
⊕ logical exclusive OR
^ logical AND
v logical OR
~ logical complement

* affected
– unaffected
0 cleared
1 set
U undefined

Preliminary

Instruction Prefetch

The SCN68010 uses a two-word tightly-coupled instruction prefetch mechanism to enhance performance. This mechanism is described in terms of the microcode operations involved. If the execution of an instruction is defined to begin when the microroutine for that instruction is entered, some features of the prefetch mechanism can be described.

1. When execution of an instruction begins, the operation word and the word following have already been fetched. The operation word is in the instruction decoder.
2. In the case of multi-word instructions, as each additional word of the instruction is used internally, a fetch is made to the instruction stream to replace it.
3. The last fetch for an instruction from the instruction stream is made when the operation word is discarded and decoding is started on the next instruction.
4. If the instruction is a single-word instruction causing a branch, the second word is not used. But because this word is fetched by the preceding instruction, it is impossible to avoid this superfluous fetch.
5. In the case of an interrupt or trace exception, both prefetched words are not used.
6. The program counter usually points to the last word fetched from the instruction stream.

Loop Mode Operation

The SCN68010 has several features that provide efficient execution of program loops. One of these features is the DBcc looping primitive instruction. The DBcc instruction operates on three operands, a loop counter, a branch condition, and a branch displacement. When the DBcc is executed in loop mode, the contents of the low order word of the register specified as the loop counter is decremented by one and compared to minus one. If equal to minus one, the result of the decrement is placed back into the count register and the next sequential instruction is executed, otherwise the condition code register is checked against the specified branch condition. If the condition is true, the result of the decrement is discarded and the next sequential instruction is executed. Finally, if the count register is not equal to minus one and the branch condition is false, the branch displacement is added to the program counter and instruction execution continues at that new address. Note that this is slightly different than non-looped execution; however, the results are the same.

```
        LEA       SOURCE, A0      Load a pointer to source data
        LEA       DEST, A1        Load a pointer to destination
        MOVE. W   #LENGTH, D0     Load the counter register
LOOP    MOVE. W   (A0)+, (A1)+    Loop to move the block of data
        DBEQ      D0, LOOP        Stop if data word is zero
```

Figure 46. DBcc Loop Program Example

An example of using the DBcc instruction in a simple loop for moving a block of data is shown in figure 46. In this program, the block of data 'LENGTH' words long at address 'SOURCE' is to be moved to address 'DEST' provided that none of the words moved are equal to zero. When the effect of instruction prefetch on this loop is examined it can be seen that the bus activity during the loop execution would be:

1. Fetch the MOVE.W instruction,
2. Fetch the DBEQ instruction,
3. Read the operand where A0 points,
4. Write the operand where A1 points,
5. Fetch the DBEQ branch displacement, and
6. If loop conditions are met, return to step 1.

During this loop, five bus cycles are executed; however, only two bus cycles perform the data movement. Since the SCN68010 has a two word prefetch queue in addition to a one word instruction decode register, it is evident that the three instruction fetches in this loop could be eliminated by placing the MOVE.W word in the instruction decode register and holding the DBEQ instruction and its branch displacement in the prefetch queue. The SCN68010 has the ability to do this by entering the loop mode of operation. During loop mode operation, all opcode fetches are suppressed and only operand reads and writes are performed until an exit loop condition is met.

Loop mode operation is transparent to the programmer, with only two conditions required for the SCN68010 to enter the loop mode. First, a DBcc instruction must be executed with both branch conditions met and a branch displacement of minus four; which indicates that the branch is to a one word instruction preceding the DBcc instruction. Second, when the processor fetches the instruction at the branch address, it is checked to determine whether it is one of the allowed looping instructions. If it is, the loop mode is entered. Thus, the single word looped instruction and the first word of the DBcc instruction will each be fetched twice when the loop is entered; but no instruction fetches will occur again until the DBcc loop conditions fail.

In addition to the normal termination conditions for a loop, there are several conditions that will cause the SCN68010 to exit loop mode operation. These conditions are interrupts, trace exceptions, reset errors, and bus errors. Interrupts are honored after each execution of the DBcc instruction, but not after the execution of the looped instruction. If an interrupt exception occurs, loop mode operation is terminated and can be restarted on return from the interrupt handler. If the T bit is set, trace exceptions will occur at the end of both the loop instruction and the DBcc instruction and thus loop mode operation is not available. Reset will abort all processing, including the loop mode. Bus errors during the loop mode will be treated the same as in normal processing; however, when the RTE instruction is used to continue the execution of the looped instruction, the three word loop will not be re-fetched.

The loopable instructions available on the SCN68010 are listed in table 24. These instructions may use the three address register indirect addressing modes to form one word looping instructions; (An), (An) + , and −(An).

Preliminary

Table 24. MC68010 LOOPABLE INSTRUCTIONS

OPCODES	APPLICABLE ADDRESSING MODES
MOVE [BWL]	(Ay) to (Ax) (Ay) to (Ax)+ (Ay) to -(Ax) (Ay)+ to (Ax) (Ay)+ to (Ax)+ (Ay)+ to -(Ax) -(Ay) to (Ax) -(Ay) to (Ax)+ -(Ay) to -(Ax) Ry to (Ax) Ry to (Ax)+
ADD [BWL] AND [BWL] CMP [BWL] OR [BWL]. SUB [BWL]	(Ay) to Dx (Ay)+ to Dx -(Ay) to Dx
ADDA [WL] CMPA [WL] SUBA [WL]	(Ay) to Ax -(Ay) to Ax (Ay)+ to Ax
ADD [BWL] AND [BWL] EOR [BWL] OR [BWL] SUB [BWL]	Dx to (Ay) Dx to (Ay)+ Dx to -(Ay)

OPCODES	APPLICABLE ADDRESSING MODES
ABCD [B] ADDX [BWL] SBCD [B] SUBX [BWL]	-(Ay) to -(Ax)
CMP [BWL]	(Ay)+ to (Ax)+
CLR [BWL] NEG [BWL] NEGX [BWL] NOT [BWL] TST [BWL] NBCD [B]	(Ay) (Ay)+ -(Ay)
ASL [W] ASR [W] LSL [W] LSR [W] ROL [W] ROR [W] ROXL [W] ROXR [W]	(Ay) by #1 (Ay)+ by #1 -(Ay) by #1

NOTE:
[B, W, or L] indicate an operand size of byte, word, or long word.

Instruction Execution Times

The following paragraphs contain listings of the instruction execution times in terms of external clock (CLK) periods. In this timing data, it is assumed that both memory read and write cycle times are four clock periods. Any wait states caused by a longer memory cycle must be added to the total instruction time. The number of bus read and write cycles for each instruction is also included with the timing data. This data is enclosed in parenthesis following the execution periods and is shown as (r/w) where r is the number of read cycles and w is the number of write cycles.

NOTE
The number of clock periods includes instruction fetches and all applicable operand fetches and stores.

Operand Effective Address Calculation Times

Table 25 lists the number of clock periods required to compute an instruction's effective address. It includes fetching of any extension words, the address computation, and fetching of the memory operand if necessary. Several instructions do not need the operand at an effective address to be fetched and thus require fewer clock periods to calculate a given effective address than the instructions that do fetch the effective address operand. The number of bus read and write cycles is shown in parentheses as (r/w). Note there are no write cycles involved in processing the effective address.

Table 25. EFFECTIVE ADDRESS CALCULATION TIMES

ADDRESSING MODE		BYTE, WORD		LONG	
		Fetch	No Fetch	Fetch	No Fetch
	Register				
Dn	Data register direct	0(0/0)	-	0(0/0)	-
An	Address register direct	0(0/0)	-	0(0/0)	-
	Memory				
(An)	Address register indirect	4(1/0)	2(0/0)	8(2/0)	2(0/0)
(An)+	Address register indirect with postincrement	4(1/0)	4(0/0)	8(2/0)	4(0/0)
-(An)	Address register indirect with predecrement	6(1/0)	4(0/0)	10(2/0)	4(0/0)
d(An)	Address register indirect with displacement	8(2/0)	4(0/0)	12(3/0)	4(1/0)
d(An, ix)*	Address register indirect with index	10(2/0)	8(1/0)	14(3/0)	8(1/0)
xxx.W	Absolute short	8(2/0)	4(1/0)	12(3/0)	4(1/0)
xxx.L	Absolute long	12(3/0)	8(2/0)	16(4/0)	8(2/0)
d(PC)	Program counter with displacement	8(2/0)	-	12(3/0)	-
d(PC, ix)	Program counter with index	10(2/0)	-	14(3/0)	-
#xxx	Immediate	4(1/0)	-	8(2/0)	-

*The size of the index register (ix) does not affect execution time.

Preliminary

Table 26. MOVE BYTE AND WORD INSTRUCTION EXECUTION TIMES

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	d(An)	d(An, ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
An	4(1/0)	4(1/0)	8(1/1)	8(1/1)	8(1/1)	12(2/1)	14(2/1)	12(2/1)	16(3/1)
(An)	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
(An)+	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)
-(An)	10(2/0)	10(2/0)	14(2/1)	14(2/1)	14(2/1)	18(3/1)	20(3/1)	18(3/1)	22(4/1)
d(An)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
d(An, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
xxx.W	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
xxx.L	16(4/0)	16(4/0)	20(4/1)	20(4/1)	20(4/1)	24(5/1)	26(5/1)	24(5/1)	28(6/1)
d(PC)	12(3/0)	12(3/0)	16(3/1)	16(3/1)	16(3/1)	20(4/1)	22(4/1)	20(4/1)	24(5/1)
d(PC, ix)*	14(3/0)	14(3/0)	18(3/1)	18(3/1)	18(3/1)	22(4/1)	24(4/1)	22(4/1)	26(5/1)
#xxx	8(2/0)	8(2/0)	12(2/1)	12(2/1)	12(2/1)	16(3/1)	18(3/1)	16(3/1)	20(4/1)

*The size of the index register (ix) does not affect execution time.

Move Instruction Execution Times

Tables 26, 27, 28, and 29 indicate the number of clock periods for the move instruction. This data includes instruction fetch, operand reads, and operand writes. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 27. MOVE BYTE AND WORD INSTRUCTION LOOP MODE EXECUTION TIMES

	LOOP CONTINUED			LOOP TERMINATED					
	Valid Count, cc False			Valid Count, cc True			Expired Count		
	Destination								
SOURCE	(An)	(An)+	-(An)	(An)	(An)+	-(An)	(An)	(An)+	-(An)
Dn	10(0/1)	10(0/1)	-	18(2/1)	18(2/1)	-	16(2/1)	16(2/1)	-
An*	10(0/1)	10(0/1)	-	18(2/1)	18(2/1)	-	16(2/1)	16(2/1)	-
(An)	14(1/1)	14(1/1)	16(1/1)	20(3/1)	20(3/1)	22(3/1)	18(3/1)	18(3/1)	20(3/1)
(An)+	14(1/1)	14(1/1)	16(1/1)	20(3/1)	20(3/1)	22(3/1)	18(3/1)	18(3/1)	20(3/1)
-(An)	16(1/1)	16(1/1)	18(1/1)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)

*Word only.

Table 28. MOVE LONG INSTRUCTION EXECUTION TIMES

SOURCE	DESTINATION								
	Dn	An	(An)	(An)+	-(An)	d(An)	d(An, ix)*	xxx.W	xxx.L
Dn	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
An	4(1/0)	4(1/0)	12(1/2)	12(1/2)	14(1/2)	16(2/2)	18(2/2)	16(2/2)	20(3/2)
(An)	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
(An)+	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)
-(An)	14(3/0)	14(3/0)	22(3/2)	22(3/2)	22(3/2)	26(4/2)	28(4/2)	26(4/2)	30(5/2)
d(An)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
d(An, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
xxx.W	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(6/2)
xxx.L	20(5/0)	20(5/0)	28(5/2)	28(5/2)	28(5/2)	32(6/2)	34(6/2)	32(6/2)	36(7/2)
d(PC)	16(4/0)	16(4/0)	24(4/2)	24(4/2)	24(4/2)	28(5/2)	30(5/2)	28(5/2)	32(5/2)
d(PC, ix)*	18(4/0)	18(4/0)	26(4/2)	26(4/2)	26(4/2)	30(5/2)	32(5/2)	30(5/2)	34(6/2)
#xxx	12(3/0)	12(3/0)	20(3/2)	20(3/2)	20(3/2)	24(4/2)	26(4/2)	24(4/2)	28(5/2)

*The size of the index register (ix) does not affect execution time.

Preliminary

Table 29. MOVE LONG INSTRUCTION LOOP MODE EXECUTION TIMES

	LOOP CONTINUED			LOOP TERMINATED					
	Valid Count, cc False			Valid Count, cc True			Expired Count		
	Destination								
SOURCE	(An)	(An)+	-(An)	(An)	(An)+	-(An)	(An)	(An)+	-(An)
Dn	14(0/2)	14(0/2)	-	20(2/2)	20(2/2)	-	18(2/2)	18(2/2)	-
An	14(0/2)	14(0/2)	-	20(2/2)	20(2/2)	-	18(2/2)	18(2/2)	-
(An)	22(2/2)	22(2/2)	24(2/2)	28(4/2)	28(4/2)	30(4/2)	24(4/2)	24(4/2)	26(4/2)
(An)+	22(2/2)	22(2/2)	24(2/2)	28(4/2)	28(4/2)	30(4/2)	24(4/2)	24(4/2)	26(4/2)
-(An)	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	26(4/2)	26(4/2)	28(4/2)

Standard Instruction Execution Times

The number of clock periods shown in tables 30 and 31 indicate the time required to perform the operations, store the results, and read the next instruction. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In tables 30 and 31 the headings have the following meanings: An = address register operand, Dn = data register operand, ea = an operand specified by an effective address, and M = memory effective address operand.

Table 30. STANDARD INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op <ea>, An***	op <ea>, Dn	op Dn, <M>
ADD	byte, word	8(1/0)+	4(1/0)+	8(1/1)+
	long	6(1/0)+	6(1/0)	12(1/2)+
AND	byte, word	-	4(1/0)+	8(1/1)+
	long	-	6(1/0)+	12(1/2)+
CMP	byte, word	6(1/0)+	4(1/0)+	-
	long	6(1/0)+	6(1/0)+	-
DIVS	-	-	122(1/0)+	-
DIVU	-	-	108(1/0)+	-
EOR	byte, word	-	4(1/0)+**	18(1/1)+
	long	-	6(1/0)**	12(1/2)+
MULS	-	-	42(1/0)+*	-
MULU	-	-	40(1/0)+	-
OR	byte, word	-	4(1/0)+	8(1/1)+
	long	-	6(1/0)+	12(1/2)+
SUB	byte, word	8(1/0)+	4(1/0)+	8(1/1)+
	long	6(1/0)+	6(1/0)+	12(1/2)+

NOTES:
+ add effective address calculation time
* indicates maximum value
** only available addressing mode is data register direct
*** word or long only

Preliminary

Table 31. STANDARD INSTRUCTION LOOP MODE EXECUTION TIMES

		LOOP CONTINUED			LOOP TERMINATED					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
INSTRUCTION	SIZE	op <ea>, An*	op <ea>, Dn	op Dn, <ea>	op <ea>, An*	op <ea>, Dn	op Dn, <ea>	op <ea>, An*	op <ea>, Dn	op Dn, <ea>
ADD	byte, word	18(1/0)	16(1/0)	16(1/1)	24(3/0)	22(3/0)	22(3/1)	22(3/0)	20(3/0)	20(3/1)
	long	22(2/0)	22(2/0)	24(2/2)	28(4/0)	28(4/0)	30(4/2)	26(4/0)	26(4/0)	28(4/2)
AND	byte, word	–	16(1/0)	16(1/1)	–	22(3/0)	22(3/1)	–	20(3/0)	20(3/1)
	long	–	22(2/0)	24(2/2)	–	28(4/0)	30(4/2)	–	26(4/0)	28(4/2)
CMP	byte, word	12(1/0)	12(1/0)	–	18(3/0)	18(3/0)	–	16(3/0)	16(4/0)	–
	long	18(2/0)	18(2/0)	–	24(4/0)	24(4/0)	–	20(4/0)	20(4/0)	–
EOR	byte, word	–	–	16(1/0)	–	–	22(3/1)	–	–	20(3/1)
	long	–	–	24(2/2)	–	–	30(4/2)	–	–	28(4/2)
OR	byte, word	–	16(1/0)	16(1/0)	–	22(3/0)	22(3/1)	–	20(3/0)	20(3/1)
	long	–	22(2/0)	24(2/2)	–	28(4/0)	30(4/2)	–	26(4/0)	28(4/2)
SUB	byte, word	18(1/0)	16(1/0)	16(1/1)	24(3/0)	22(3/0)	22(3/1)	22(3/0)	20(3/0)	20(3/1)
	long	22(2/0)	20(2/0)	24(2/2)	28(4/0)	26(4/0)	(30(4/2)	26(4/0)	24(4/0)	28(4/2)

*Word or long only.
<ea> may be (An), +(An), or –(An) only. Add two clock periods to the table value if <ea> is –(An).

Immediate Instruction Execution Times

The number of clock periods shown in table 32 includes the time to fetch immediate operands, perform the operations, store the results, and read the next operation. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

In table 32, the headings have the following meanings: # = immediate operand, Dn = data register operand, AN = address register operand, and M = memory operand.

Table 32. IMMEDIATE INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	op #, Dn	op #, An	op #, M
ADDI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
ADDQ	byte, word	4(1/0)	4(1/0)*	8(1/1)+
	long	8(1/0)	8(1/0)	12(1/2)+
ANDI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/1)+
CMPI	byte, word	8(2/0)	–	8(2/0)+
	long	12(3/0)	–	12(3/0)+
EORI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
MOVEQ	long	4(1/0)	–	–
ORI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
SUBI	byte, word	8(2/0)	–	12(2/1)+
	long	14(3/0)	–	20(3/2)+
SUBQ	byte, word	4(1/0)	4(1/0)*	8(1/1)+
	long	8(1/0)	8(1/0)	12(1/2)+

+ add effective address calculation time
* word only

Preliminary

Single Operand Instruction Execution Times

Tables 33, 34, and 35 indicate the number of clock periods for the single operand instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 33. SINGLE OPERAND INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY
NBCD	byte	6(1/0)	8(1/1)+
NEG	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NEGX	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
NOT	byte, word	4(1/0)	8(1/1)+
	long	6(1/0)	12(1/2)+
S_{CC}	byte, false	4(1/0)	8(1/1)+ *
	byte, true	4(1/0)	8(1/1)+ *
TAS	byte	4(1/0)	14(2/1)+ *
TST	byte, word	4(1/0)	4(1/0)
	long	4(1/0)	4(1/0)+

+ add effective address calculation time
* Use non-fetching effective address calculation time.

Table 34. CLEAR INSTRUCTION EXECUTION TIMES

	SIZE	Dn	An	(An)	(An)+	-(An)	d(An)	(An, ix)*	xxx.W	xxx.L
CLR	byte, word	4(1/0)	–	8(1/1)	8(1/1)	10(1/1)	12(2/1)	16(2/1)	12(2/1)	16(3/1)
	long	6(1/0)	–	12(1/2)	12(1/2)	14(1/2)	16(2/2)	20(2/2)	16(2/2)	20(3/2)

* The size of the index register (ix) does not affect execution time.

Table 35. SINGLE OPERAND INSTRUCTION LOOP MODE EXECUTION TIMES

		LOOP CONTINUED			LOOP TERMINATED					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
INSTRUCTION	SIZE	(An)	(An)+	-(An)	(An)	(An)+	-(An)	(An)	(An)+	-(An)
CLR	byte, word	10(0/1)	10(0/1)	12(0/1)	18(2/1)	18(2/1)	20(2/0)	16(2/1)	16(2/1)	18(2/1)
	long	14(0/2)	14(0/2)	16(0/2)	22(2/2)	22(2/2)	24(2/2)	20(2/2)	20(2/2)	22(2/2)
NBCD	byte	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
NEG	byte, word	16(1/1)	16(1/1)	18(2/2)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)
	long	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	28(4/2)	28(4/2)	30(4/2)
NEGX	byte, word	16(1/1)	16(1/1)	18(2/2)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)
	long	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	28(4/2)	28(4/2)	30(4/2)
NOT	byte, word	16(1/1)	16(1/1)	18(2/2)	22(3/1)	22(3/1)	24(3/1)	20(3/1)	20(3/1)	22(3/1)
	long	24(2/2)	24(2/2)	26(2/2)	30(4/2)	30(4/2)	32(4/2)	28(4/2)	28(4/2)	30(4/2)
TST	byte, word	12(1/0)	12(1/0)	14(1/0)	18(3/0)	18(3/0)	20(3/0)	16(3/0)	16(3/0)	18(3/0)
	long	18(2/0)	18(2/0)	20(2/0)	24(4/0)	24(4/0)	26(4/0)	20(4/0)	20(4/0)	22(4/0)

Preliminary

Shift/Rotate Instruction Execution Times

Tables 36 and 37 indicate the number of clock periods for the shift and rotate instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 36. SHIFT/ROTATE INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY*
ASR, ASL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
LSR, LSL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
ROR, ROL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–
ROXR, ROXL	byte, word	6 + 2n(1/0)	8(1/1)+
	long	8 + 2n(1/0)	–

+ add effective address calculation time
n is the shift or rotate count
* word only

Table 37. SHIFT/ROTATE INSTRUCTION LOOP MODE EXECUTION TIMES

		LOOP CONTINUED			LOOP TERMINATED					
		Valid Count, cc False			Valid Count, cc True			Expired Count		
INSTRUCTION	SIZE	(An)	(An)+	–(An)	(An)	(An)+	–(An)	(An)	(An)+	–(An)
ASR, ASL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
LSR, LSL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
ROR, ROL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)
ROXR, ROXL	word	18(1/1)	18(1/1)	20(1/1)	24(3/1)	24(3/1)	26(3/1)	22(3/1)	22(3/1)	24(3/1)

Bit Manipulation Instruction Execution Times

Table 38 indicates the number of clock periods required for the bit manipulation instructions. The number of bus read and write cycles is shown in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

Table 38. BIT MANIPULATION INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	DYNAMIC		STATIC	
		Register	Memory	Register	Memory
BCHG	byte	–	8(1/1)+	–	12(2/1)+
	long	8(1/0)*	–	12(2/0)*	–
BCLR	byte	–	10(1/1)+	–	14(2/1)+
	long	10(1/0)*	–	14(2/0)*	–
BSET	byte	–	8(1/0)+	–	12(2/1)+
	long	8(1/0)*	–	12(2/0)*	–
BTST	byte	–	4(1/0)+	–	8(2/0)+
	long	6(1/0)*	–	10(2/0)	–

+ add effective address calculation time
* indicates maximum value

Preliminary

Conditional Instruction Execution Times
Table 39 indicates the number of clock periods required for the conditional instructions. The number of bus read and write cycles is indicated in parenthesis as (r/w). The number of clock periods and the number of read and write cycles must be added respectively to those of the effective address calculation where indicated.

JMP, JSR, LEA, PEA, and MOVEM Instruction Execution Times
Table 40 indicates the number of clock periods required for the jump, jump-to-subroutine, load effective address, push effective address, and move multiple registers instructions. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 39. CONDITIONAL INSTRUCTION EXECUTION TIMES

INSTRUCTION	DISPLACEMENT	BRANCH TAKEN	BRANCH NOT TAKEN
B_{CC}	byte	10(2/0)	6(1/0)
	word	10(2/0)	10(2/0)
BRA	byte	10(2/0)	–
	word	10(2/0)	–
BSR	byte	18(2/2)	–
	word	18(2/2)	–
DB_{CC}	CC true	–	10(2/0)
	CC false	10(2/0)	16(3/0)

Table 40. JMP, JSR, LEA, PEA, AND MOVEM INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	(An)	(An)+	-(An)	d(An)	d(An, ix)+	xxx.W	xxx.L	d(PC)	d(PC, ix)*
JMP	–	8(2/0)	–	–	10(2/0)	14(3/0)	10(2/0)	12(3/0)	10(2/0)	14(3/0)
JSR	–	16(2/2)	–	–	18(2/2)	22(2/2)	18(2/2)	20(3/2)	18(2/2)	22(2/2)
LEA	–	4(1/0)	–	–	8(2/0)	12(2/0)	8(2/0)	12(3/0)	8(2/0)	12(2/0)
PEA	–	12(1/2)	–	–	16(2/2)	20(2/2)	16(2/2)	20(3/2)	16(2/2)	20(2/2)
MOVEM M → R	word	12 + 4n (3 + n/0)	12 + 4n (3 + n/0)	–	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)	16 + 4n (4 + n/0)	20 + 4n (5 + n/0)	16 + 4n (4 + n/0)	18 + 4n (4 + n/0)
	long	12 + 8n (3 + 2n/0)	12 + 8n (3 + 2n/0)	–	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)	16 + 8n (4 + 2n/0)	20 + 8n (5 + 2n/0)	16 + 8n (4 + 2n/0)	18 + 8n (4 + 2n/0)
MOVEM R → M	word	8 + 4n (2/n)	–	8 + 4n (2/n)	12 + 4n (3/n)	14 + 4n (3/n)	12 + 4n (3/n)	16 + 4n (4/n)	– –	– –
	long	8 + 8n (2/2n)	– –	8 + 8n (2/2n)	12 + 8n (3/2n)	14 + 8n (3/2n)	12 + 8n (3/2n)	16 + 8n (4/2n)	– –	– –

n is the number of registers to move
* The size of the index register (ix) does not affect the instruction's execution time

Multi-Precision Instruction Execution Times
Table 41 indicates the number of clock periods for the multi-precision instructions. The number of clock periods includes the time to fetch both operands, perform the operations, store the results, and read the next instructions. The number of read and write cycles is shown in parenthesis as (r/w).

In table 41, the headings have the following meanings: Dn = data register operand and M = memory operand.

Table 41. MULTI-PRECISION INSTRUCTION EXECUTION TIMES

		NON-LOOPED	LOOP MODE		
			Continued	Terminated	
			Valid Count, cc False	Valid Count, ccTrue	Expired Count
INSTRUCTION	SIZE	op Dn, Dn	op M, M*		
ADDX	byte, word	4(1/0)	18(3/10)	22(2/1)	28(4/1)	26(4/1)
	long	6(1/0)	30(5/2)	32(4/2)	38(6/2)	36(6/2)
CMPM	byte, word	–	12(3/0)	14(2/0)	20(4/0)	18(4/0)
	long	–	20(5/0)	24(4/0)	30(6/0)	26(6/0)
SUBX	byte, word	4(1/0)	18(3/1)	22(2/1)	28(4/1)	26(4/1)
	long	6(1/0)	30(5/2)	32(4/2)	38(6/2)	36(6/2)
ABCD	byte	6(1/0)	18(3/1)	24(2/1)	30(4/1)	28(4/1)
SBCD	byte	6(1/0)	18(3/1)	24(2/1)	30(4/1)	28(4/1)

*Source and destination ea is (An)+ for CMPM and -(An) for all others.

Preliminary

Miscellaneous Instruction Execution Times

Table 42 indicates the number of clock periods for the following miscellaneous instructions. The number of bus read and write cycle is shown in parenthesis as (r/w). The number of clock periods plus the number of read and write cycles must be added to those of the effective address calculation where indicated.

Exception Processing Execution Times

Table 43 indicates the number of clock periods for exception processing. The number of clock periods includes the time for all stacking, the vector fetch, and the fetch of the first two instruction words of the handler routine. The number of bus read and write cycles is shown in parenthesis as (r/w).

Table 43. EXCEPTION PROCESSING EXECUTION TIMES

EXCEPTION	
Address error	126(4/26)
Breakpoint instruction*	42(5/4)
Bus error	126(4/26)
CHK instruction**	44(5/4)+
Divide by zero	42(5/4)
Illegal instruction	38(4/4)
Interrupt*	46(5/4)
MOVEC, illegal cr**	46(5/4)
Privilege violation	38(4/4)
Reset***	40(6/0)
RTE, illegal format	50(7/4)
RTE, illegal revision	70(12/4)
Trace	38(4/4)
TRAP instruction	38(4/4)
TRAPV instruction	40(5/4)

+ add effective address calculation time.
*The interrupt acknowledge and breakpoint cycles are assumed to take four clock periods.
**Indicates maximum value
***Indicates the time from when $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ are first sampled as negated to when instruction execution starts.

Table 42. MISCELLANEOUS INSTRUCTION EXECUTION TIMES

INSTRUCTION	SIZE	REGISTER	MEMORY	REGISTER → DESTINATION	SOURCE** → REGISTER
ANDI to CCR	–	16(2/0)	–	–	–
ANDI to SR	–	16(2/0)	–	–	–
CHK	–	8(1/0)+	–	–	–
EORI to CCR	–	16(2/0)	–	–	–
EORI to SR	–	16(2/0)	–	–	–
EXG	–	6(1/0)	–	–	–
EXT	word	4(1/0)	–	–	–
	long	4(1/0)	–	–	–
LINK	–	16(2/2)	–	–	–
MOVE from CCR	–	4(1/0)	8(1/1)+*	–	–
MOVE to CCR	–	12(2/0)	12(2/0)+	–	–
MOVE from SR	–	4(1/0)	8(1/1)+*	–	–
MOVE to SR	–	12(2/0)	12(2/0)+	–	–
MOVE from USP	–	6(1/0)	–	–	–
MOVE to USP	–	6(1/0)	–	–	–
MOVEC	–	–	–	10(2/0)	12(2/0)
MOVEP	word	–	–	16(2/2)	16(4/0)
	long	–	–	24(2/4)	24(6/0)
NOP	–	4(1/0)	–	–	–
ORI to CCR	–	16(2/0)	–	–	–
ORI to SR	–	16(2/0)	–	–	–
RESET	–	130(1/0)	–	–	–
RTD	–	16(4/0)	–	–	–
RTE	short	24(6/0)	–	–	–
	long, retry read	112(27/10)	–	–	–
	long, retry write	112(26/1)	–	–	–
	long, no retry	110(26/0)	–	–	–
RTR	–	20(5/0)	–	–	–
RTS	–	16(4/0)	–	–	–
STOP	–	4(0/0)	–	–	–
SWAP	–	4(1/0)	–	–	–
TRAPV	–	4(1/0)	–	–	–
UNLK	–	12(3/0)	–	–	–

+ add effective address calculation time.
* use non-fetching effective address calculation time.
** Source or destination is a memory location for the MOVEP instruction and a control register for the MOVEC instruction.

Preliminary

ABSOLUTE MAXIMUM RATINGS

RATING	SYMBOL	VALUE	UNIT
Supply voltage	V_{CC}	−0.3 to +7.0	V
Input voltage	V_{in}	−0.3 to +7.0	V
Operating temperature range SCN68010 SCN68010 ceramic	T_A	T_L to T_H 0 to 70 −40 to 85	°C
Storage temperature	T_{stg}	−55 to 150	°C

NOTE:
This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either V_{SS} or V_{CC}).

THERMAL CHARACTERISTICS

CHARACTERISTIC	SYMBOL	VALUE	RATING
Thermal Resistance Ceramic Plastic Chip Carrier	θ_{JA}	 30 30 50	 °C/W

Power Considerations

The average chip-junction temperature, T_J, in °C can be obtained from:

$$T_J = T_A + (P_D \bullet \theta_{JA}) \quad (1)$$

Where:

T_A = ambient temperature, °C
θ_{JA} = package thermal resistance, junction-to-ambient, °C/W
$P_D = P_{INT} + P_{I/O}$
$P_{INT} = I_{CC} \times V_{CC}$, watts — chip internal power
$P_{I/O}$ = power dissipation on input and output pins — user determined

For most applications $P_{I/O} < P_{INT}$ and can be neglected.

An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad (2)$$

Solving equations 1 and 2 for K gives:

$$K = T_D \bullet (T_A + 273°C) + \theta_{JA} \bullet P_D^2 \quad (3)$$

Where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K the values of P_D and T_J can be obtained by solving equations (1) and (2) iteratively for any value of T_A.

The curve shown in figure 47 gives the graphic solution to these equations for the specification power dissipation of 1.50 and 1.75 watts over the ambient temperature range of −55°C to 125°C using a θ_{JA} of 45°C/W for the ceramic package.

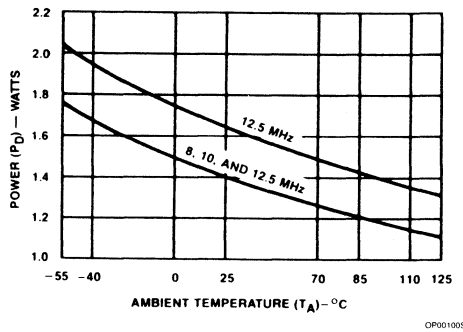


Figure 47. SCN68010 Power Dissipation (P_D) vs Ambient Temperature (T_A)

Preliminary

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0Vdc ±5%; V_{SS} = 0Vdc; $T_A = T_L$ to T_H (see figures 48, 49, and 50)

CHARACTERISTIC		SYMBOL	MIN	MAX	UNIT
Input high voltage		V_{IH}	2.0	V_{CC}	V
Input low voltage		V_{IL}	V_{SS} - 0.3	0.8	V
Input leakage current @ 5.25V	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, CLK, $\overline{IPL0}$ - $\overline{IPL2}$, $\overline{VPA}$, $\overline{HALT}$, $\overline{RESET}$	I_{IN}	-	2.5 20	μA
Three-state (off state) input current @ 2.4V/0.4V	$\overline{AS}$, A1 - A23, D0 - D15, FC0 - FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	I_{TSI}	-	20	μA
Output high voltage (I_{OH} = -400μA)	E* E**, $\overline{AS}$, A1 - A23, $\overline{BG}$, D0 - D15, FC0 - FC2, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	V_{OH}	V_{CC} - 0.75 2.4	- -	V
Output low voltage (I_{OL} = 1.6mA) (I_{OL} = 3.2mA) (I_{OL} = 5.0mA) (I_{OL} = 5.3mA)	$\overline{HALT}$ A1 - A23, $\overline{BG}$, FC0 - FC2 $\overline{RESET}$ E, $\overline{AS}$, D0 - D7, $\overline{LDS}$, R/$\overline{W}$, $\overline{UDS}$, $\overline{VMA}$	V_{OL}	- - - -	0.5 0.5 0.5 0.5	V
Power dissipation***		P_D	-	-	W
Capacitance (V_{IN} = 0V, T_A = 25°C, frequency = 1MHz)****		C_{in}	-	20.0	pF

* With external pullup resistor of 1.1 kΩ.
** Without external pullup resistor.
*** During normal operation instantaneous V_{CC} current requirements may be as high as 1.5A.
****Capacitance is periodically sampled rather than 100% tested.

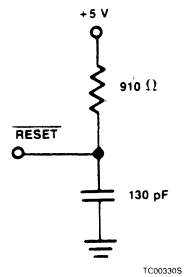


Figure 48. $\overline{RESET}$ Test Load

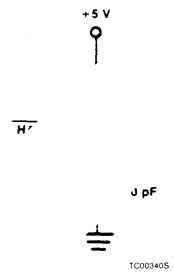


Figure 49. Test Loads

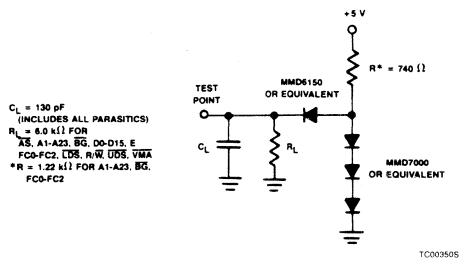


Figure 50. $\overline{HALT}$ Test Load

Preliminary

AC ELECTRICAL CHARACTERISTICS — Clock Input (See figure 51)

CHARACTERISTIC	SYMBOL	8MHz		10MHz		12.5MHz		UNIT
		Min	Max	Min	Max	Min	Max	
Frequency of operation	f	2.0	8.0	2.0	10.0	4.0	12.5	MHz
Cycle time	t_{cyc}	125	500	100	500	80	250	ns
Clock pulse width	t_{CL} t_{CH}	55 55	250 250	45 45	250 250	35 35	125 125	ns
Rise and fall times	t_{Cr} t_{Cf}	– –	10 10	– –	10 10		5 5	ns

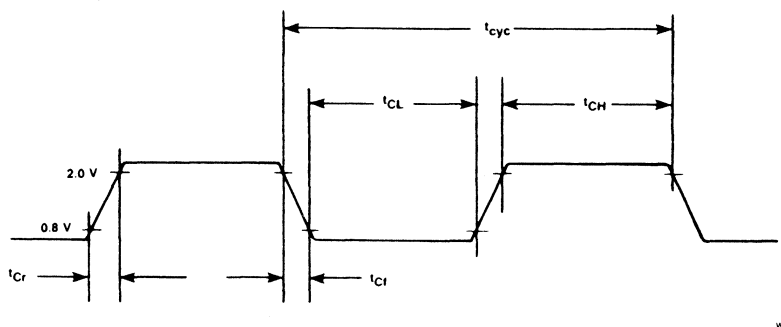


Figure 51. Clock Input Timing Diagram

Preliminary

AC ELECTRICAL CHARACTERISTICS — Read and Write Cycles (V_{CC} = 5.0Vdc ±5%; V_{SS} = 0Vdc; $T_A = T_L$ to T_H see figures 52 and 53)

NO.	CHARACTERISTIC	SYMBOL	8MHz		10MHz		12.5MHz		UNIT
			Min	Max	Min	Max	Min	Max	
1	Clock period	t_{cyc}	125	500	100	500	80	250	ns
2	Clock width low	t_{CL}	55	250	45	250	35	125	ns
3	Clock width high	t_{CH}	55	250	45	250	35	125	ns
4	Clock fall time	t_{Cf}	–	10	–	10	–	5	ns
5	Clock rise time	t_{Cr}	–	10	–	10	–	5	ns
6	Clock low to address valid	t_{CLAV}	–	70	–	55	–	55	ns
6A	Clock high to FC valid	t_{CHFCV}	–	70	–	60	–	55	ns
7	Clock high to address data high impedance (maximum)	t_{CHAZx}	–	80	–	70	–	60	ns
8	Clock high to address/FC invalid (minimum)	t_{CHAZn}	0	–	0	–	0	–	ns
9[1]	Clock high to $\overline{AS}$, $\overline{DS}$ low (maximum)	t_{CHSLx}	–	60	–	55	–	55	ns
10	Clock high to $\overline{AS}$, $\overline{DS}$ low (minimum)	t_{CHSLn}	0	–	0	–	0	–	ns
11[2]	Address valid to $\overline{AS}$, $\overline{DS}$ (read) low/$\overline{AS}$ (write)	t_{AVSL}	30	–	20	–	0	–	ns
11A[2]	FC valid to $\overline{AS}$, $\overline{DS}$ (read) low/$\overline{AS}$ (write)	t_{FCVSL}	60	–	50	–	40	–	ns
12[1]	Clock low to $\overline{AS}$, $\overline{DS}$ high	t_{CLSH}	–	70	–	55	–	50	ns
13[2]	$\overline{AS}$, $\overline{DS}$ high to address/FC invalid	t_{SHAZ}	30	–	20	–	10	–	ns
14[2]	$\overline{AS}$, $\overline{DS}$ width low (read)/$\overline{AS}$ (write)	t_{SL}	240	–	195	–	160	–	ns
14A[2]	$\overline{DS}$ width low (write)	–	115	–	95	–	80	–	ns
15[2]	$\overline{AS}$, $\overline{DS}$ width high	t_{SH}	150	–	105	–	65	–	ns
16	Clock high to $\overline{AS}$, $\overline{DS}$ high impedance	t_{CHSZ}	–	80	–	70	–	60	ns
17[2]	$\overline{AS}$, $\overline{DS}$ high to R/$\overline{W}$ high	T_{SHRH}	40	–	20	–	10	–	ns
18[1]	Clock high to R/$\overline{W}$ high (maximum)	t_{CHRHx}	–	70	–	60	–	60	ns
19	Clock high to R/$\overline{W}$ high (minimum)	t_{CHRHn}	0	–	0	–	0	–	ns
20[1]	Clock high to R/$\overline{W}$ low	t_{CHRL}	–	70	–	60	–	60	ns
20A[2]	$\overline{AS}$ low to R/$\overline{W}$ valid	t_{ASRV}	–	20	–	20	–	20	ns
21[2]	Address valid to R/$\overline{W}$ low	t_{AVRL}	20	–	0	–	0	–	ns
21A[2]	FC valid to R/$\overline{W}$ low	t_{FCVRL}	60	–	50	–	30	–	ns
22[2]	R/$\overline{W}$ low to $\overline{DS}$ low (write)	t_{RLSL}	80	–	50	–	30	–	ns
23	Clock low to data out valid	t_{CLDO}	–	70	–	55	–	55	ns
24	Clock high to R/$\overline{W}$, $\overline{VMA}$ high impedance	t_{CHRZ}	–	80	–	70	–	60	ns
25[2]	$\overline{DS}$ high to data out invalid	T_{SHDO}	30	–	20	–	15	–	ns
26[2]	Data out valid to $\overline{DS}$ low (write)	t_{DOSL}	30	–	20	–	15	–	ns
27[5]	Data in to clock low (set-up time)	t_{DICL}	15	–	10	–	10	–	ns
27A	Late $\overline{BERR}$ low to clock low (set-up time)	t_{BELCL}	45	–	45	–	45	–	ns
28[2]	$\overline{AS}$, $\overline{DS}$ high to $\overline{DTACK}$ high	t_{SHDAH}	0	245	0	190	0	150	ns
29	$\overline{DS}$ high to data invalid (hold time)	t_{SHDI}	0	–	0	–	0	–	ns
30	$\overline{AS}$, $\overline{DS}$ high to $\overline{BERR}$ high	t_{SHBEH}	0	–	0	–	0	–	ns
31[2,5]	$\overline{DTACK}$ low to data valid (set-up time)	t_{DALDI}	–	90	–	65	–	50	ns
32	$\overline{HALT}$ and $\overline{RESET}$ input transition time	$t_{RHr,\ f}$	0	200	0	200	0	200	ns
33	Clock high to $\overline{BG}$ low	t_{CHGL}	–	70	–	60	–	50	ns

Preliminary

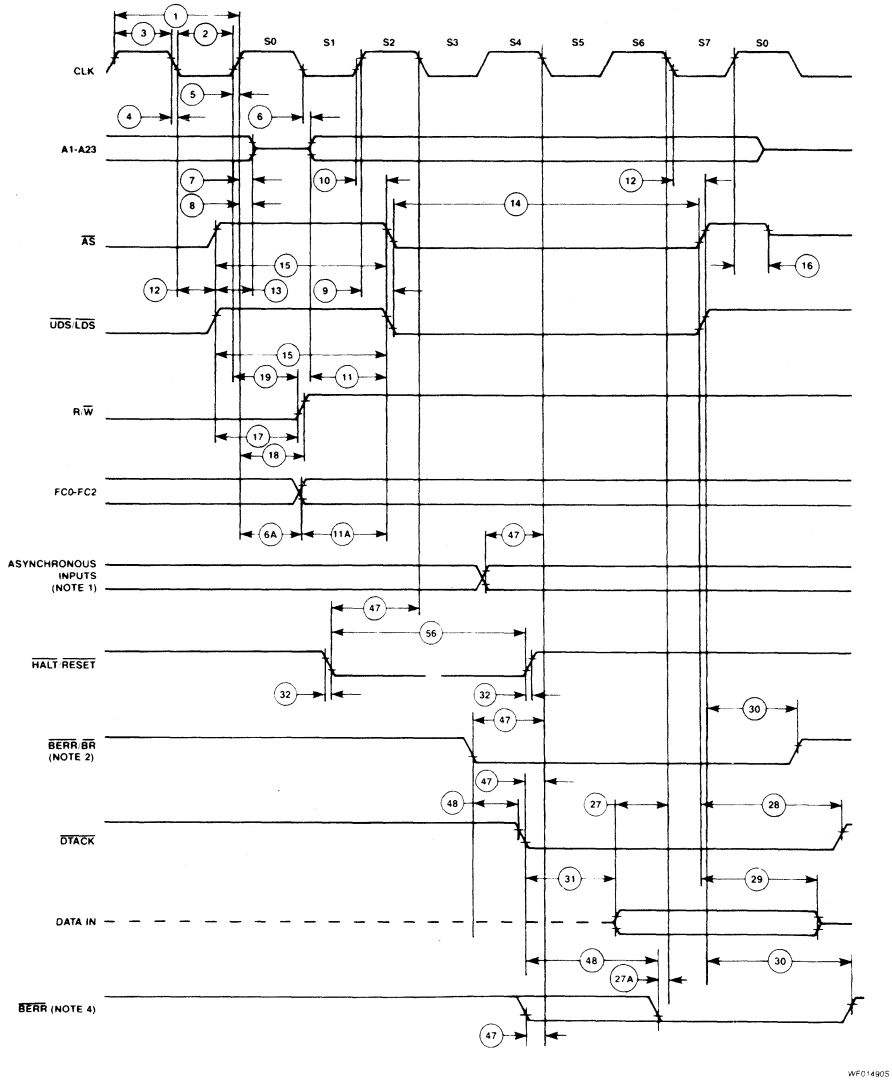
AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	CHARACTERISTIC	SYMBOL	8MHz		10MHz		12.5MHz		UNIT
			Min	Max	Min	Max	Min	Max	
34	Clock high to $\overline{BG}$ high	t_{CHGH}	–	70	–	60	–	50	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	t_{BRLGL}	1.5	3.5	1.5	3.5	1.5	3.5	clk. per.
36	$\overline{BR}$ high to $\overline{BG}$ high	t_{BRHGH}	1.5	3.5	1.5	3.5	1.5	3.5	clk. per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	t_{GALGH}	1.5	3.0	1.5	3.0	1.5	3.0	clk. per.
37A	$\overline{BGACK}$ low to $\overline{BR}$ high (to prevent rearbitration)	t_{BGKBR}	20	–	20	–	20	–	ns
38	$\overline{BG}$ low to bus high impedance (with $\overline{AS}$ high)	t_{GLZ}	–	80	–	70	–	60	ns
39	$\overline{BG}$ width high	t_{GH}	1.5	–	1.5	–	1.5	–	clk. per.
40	Clock low to $\overline{VMA}$ low	t_{CLVML}	–	70	–	70	–	70	ns
41	Clock low to E transition	t_{CLC}	–	70	–	55	–	45	ns
42	E output rise and fall time	$t_{Er, f}$	–	25	–	25	–	25	ns
43	$\overline{VMA}$ low to E high	t_{VMLEH}	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ high to $\overline{VPA}$ high	t_{SHVPH}	0	120	0	90	0	70	ns
45	E low to address / $\overline{VMA}$/ FC invalid	t_{ELAI}	30	–	10	–	10	–	ns
46	$\overline{BGACK}$ width	t_{BGL}	1.5	–	1.5	–	1.5	–	clk. per.
47[5]	Asynchronous input set-up time	t_{ASI}	20	–	20	–	20	–	ns
48[2,3]	$\overline{DTACK}$ low to $\overline{BERR}$ low	t_{DALBEL}	–	80	–	55	–	35	ns
49	E low to $\overline{AS}$, $\overline{DS}$ invalid	t_{ELSI}	−80	–	−80	–	−80	–	ns
50	E width high	t_{EH}	450	–	350	–	280	–	ns
51	E width low	t_{EL}	700	–	550	–	440	–	ns
52	E extended rise time	t_{CIEHX}	–	80	–	80	–	80	ns
53	Data hold from clock high	t_{CHDO}	0	–	0	–	0	–	ns
54	Data hold from E low (write)	t_{ELDOZ}	30	–	20	–	15	–	ns
55	R/$\overline{W}$ to data bus impedance change	t_{RLDO}	30	–	20	–	10	–	ns
56[4]	$\overline{HALT}$/$\overline{RESET}$ pulse width	t_{HRPW}	10	–	10	–	10	–	clk. per.

NOTES:

1. For a loading capacitance of less than or equal to 50 picofarads, subtract 5 nanoseconds from the values given in these columns.
2. Actual value depends on clock period.
3. In the absence of $\overline{DTACK}$, $\overline{BERR}$ is an asynchronous input using the asynchronous input set-up time (#47).
4. For power up the MPU must be held in $\overline{RESET}$ state for 100ms to allow stabilization of on-chip circuitry. After the system is powered up, #56 refers to the minimum pulse width required to reset the system.
5. If the asynchronous set-up time (#47) requirements are satisfied, the $\overline{DTACK}$-low to data set-up time (#31) and $\overline{DTACK}$–low to $\overline{BERR}$–low set-up time (#48) requirements can be ignored. The data must only satisfy the data-in to clock-low set-up time (#27) for the following clock cycle, $\overline{BERR}$ must only satisfy the late-$\overline{BERR}$-low to clock-low set-up time (#27A) for the following clock cycle.

Preliminary

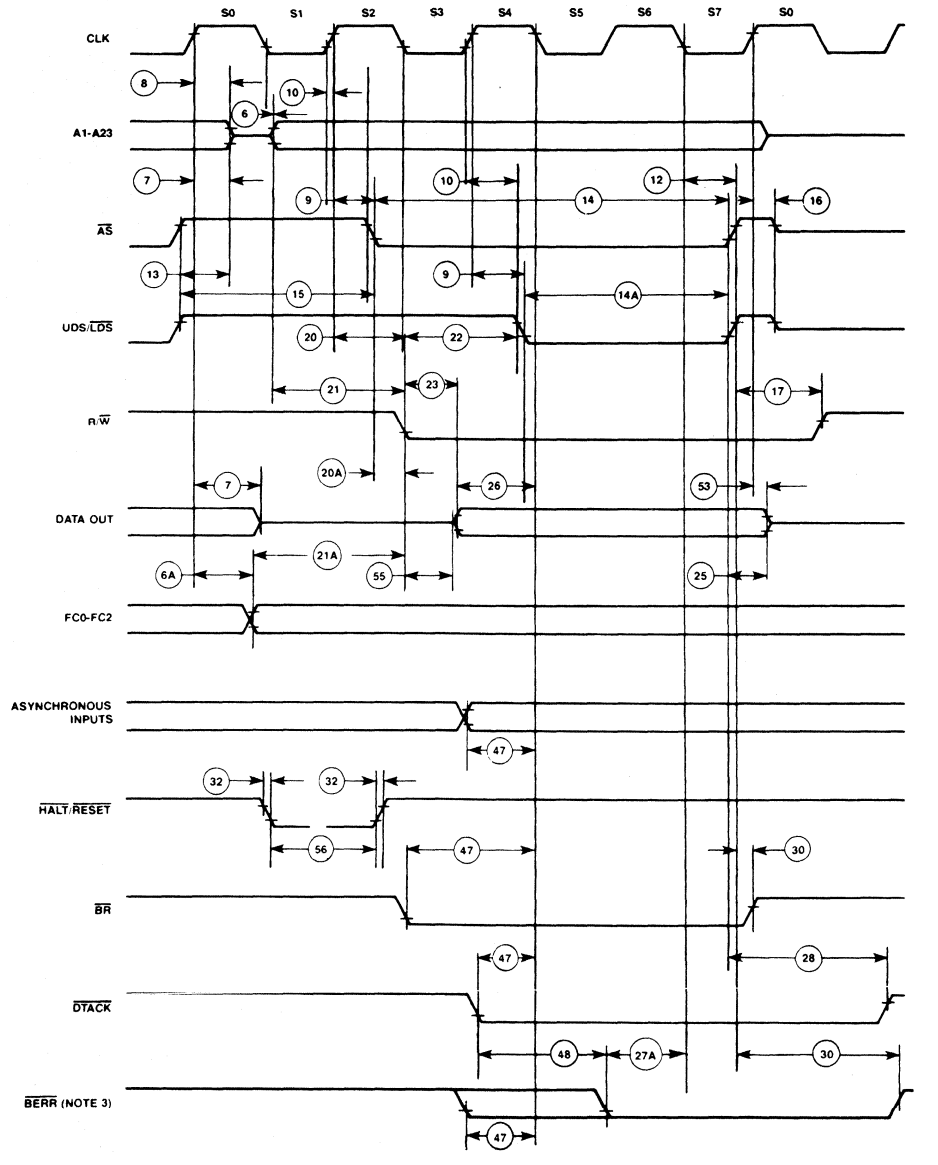


NOTES:

1. Set-up time for the asynchronous inputs $\overline{IPL0/2}$, $\overline{IPL1}$, and $\overline{VPA}$ guarantees their recognition at the next falling edge of the clock.
2. $\overline{BR}$ need fall at this time only in order to insure being recognized at the end of this bus cycle.
3. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
4. The timing for the first falling edge (47) of $\overline{BERR}$ are for $\overline{BERR}$ without $\overline{DTACK}$, the timings for the second falling edge (27A and 48) are for $\overline{BERR}$ with $\overline{DTACK}$.

Figure 52. Read Cycle Timing Diagram

Preliminary



NOTES:
1. Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts, unless otherwise noted.
2. Because of loading variations, R/$\overline{W}$ may be valid after $\overline{AS}$ even though both are initiated by the rising edge of S2 (specification 20A).
3. The timing for the first falling edge (47) of $\overline{BERR}$ are for $\overline{BERR}$ without $\overline{DTACK}$; the timings for the second falling edge (27A and 48) are for $\overline{BERR}$ with $\overline{DTACK}$.

Figure 53. Write Cycle Timing Diagram

Preliminary

AC ELECTRICAL CHARACTERISTICS — SCN68010 to Synchronous Peripheral Cycles

V_{CC} = 5.0Vdc ±5%, V_{SS} = 0Vdc, $T_A = T_L$ to T_H (refer to figures 54 and 55)

NO.	CHARACTERISTIC	SYMBOL	8MHz		10MHz		12.5MHz		UNIT
			Min	Max	Min	Max	Min	Max	
23	Clock low to data out valid	t_{CLDO}	–	70	–	55	–	55	ns
24	Clock high to R/$\overline{W}$, $\overline{VMA}$ high impedance	t_{CHRZ}	–	80	–	70	–	60	ns
27	Data in to clock low (set-up time)	t_{DICL}	15	–	10	–	10	–	ns
40	Clock low to $\overline{VMA}$ low	t_{CLVML}	–	70	–	70	–	70	ns
41	Clock low to E transition	t_{CLC}	–	70	–	55	–	45	ns
42	E output rise and fall time	$t_{Er,Ef}$	–	25	–	25	–	25	ns
43	$\overline{VMA}$ low to high	t_{VMLEH}	200	–	150	–	90	–	ns
44	$\overline{AS}$, $\overline{DS}$ high to $\overline{VPA}$ high	t_{SHVPH}	0	120	0	90	0	70	ns
45	E low to address/FC invalid	t_{ELAI}	30	–	10	–	10	–	ns
47	Asynchronous input set-up time	t_{ASI}	20	–	20	–	20	–	ns
49	E low to $\overline{AS}$, $\overline{DS}$ invalid	t_{ELSI}	-80	–	-80	–	-80	–	ns
50	E width high	t_{EH}	450	–	350	–	280	–	ns
51	E width low	t_{EL}	700	–	550	–	440	–	ns
52	E extended rise time	t_{CIEHX}	–	80	–	80		80	ns
54	Data hold from E low (write)	t_{ELDOZ}	30	–	20	–	15	–	ns

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

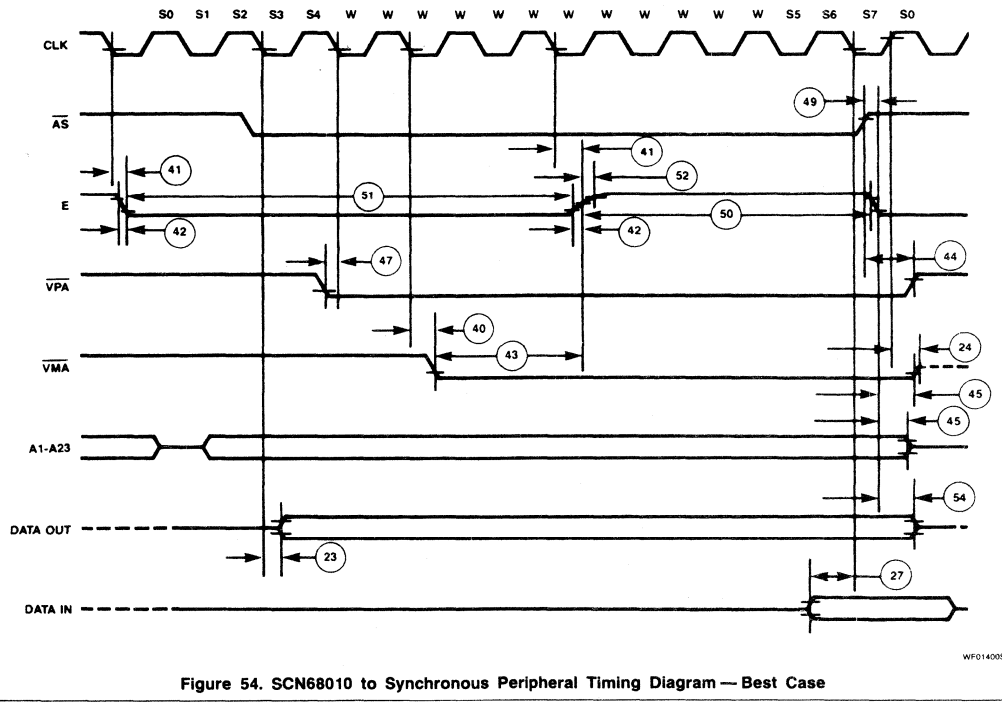


Figure 54. SCN68010 to Synchronous Peripheral Timing Diagram — Best Case

Preliminary

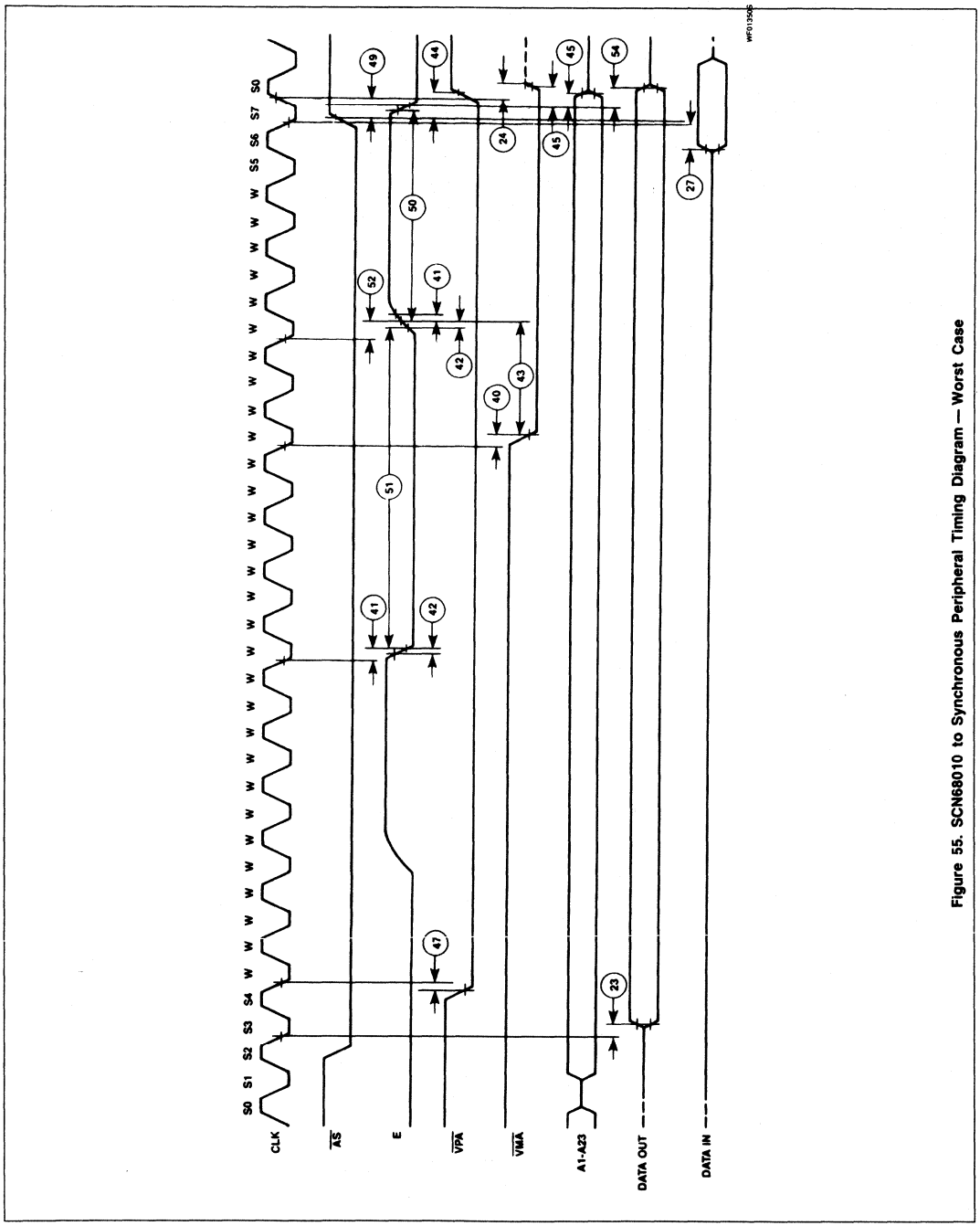


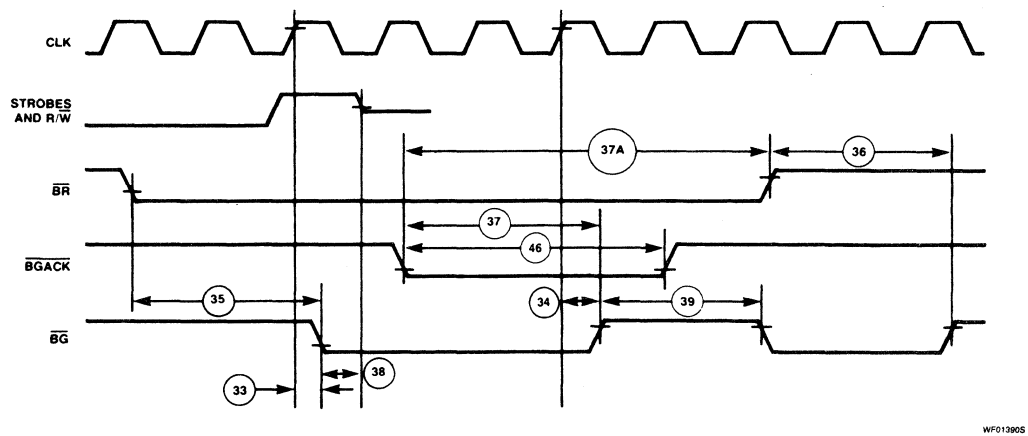
Figure 55. SCN68010 to Synchronous Peripheral Timing Diagram — Worst Case

Preliminary

AC ELECTRICAL CHARACTERISTICS — Bus Arbitration V_{CC}=5.0Vdc ±5%; V_{SS} = 0Vdc; $T_A = T_L$ to T_H (see figure 56)

NO.	CHARACTERISTIC	SYMBOL	8MHz		10MHz		12.5MHz		UNIT
			Min	Max	Min	Max	Min	Max	
33	Clock high to $\overline{BG}$ low	t_{CHGL}	–	70	–	60	–	50	ns
34	Clock high to $\overline{BG}$ high	t_{CHGH}	–	70	–	60	–	50	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	t_{BRLGL}	1.5	3.5	1.5	3.5	1.5	3.5	Clk. Per.
36	$\overline{BR}$ high to $\overline{BG}$ high	t_{BRHGH}	1.5	3.5	1.5	3.5	1.5	3.5	Clk. Per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	t_{GALGH}	1.5	3.0	1.5	3.0	1.5	3.0	Clk. Per.
37A	$\overline{BGACK}$ low to $\overline{BR}$ high (to prevent rearbitration)	t_{BGKBR}	20	–	20	–	20	–	ns
38	$\overline{BG}$ low to bus high impedance (with $\overline{AS}$ high)	t_{GLZ}	–	80	–	70	–	60	ns
39	$\overline{BG}$ width high	t_{GH}	1.5	–	1.5	–	1.5	–	Clk. Per
46	$\overline{BGACK}$ width	t_{BGL}	1.5	–	1.5	–	1.5	–	Clk. Per.

These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.



1. Set-up time for the asynchronous inputs $\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{IPL0}$-$\overline{IPL2}$, and $\overline{VPA}$ guarantees their recognition at the falling edge of the clock.
2. Waveform measurements for all inputs and outputs are specified at: logic high = 2.0 volts, logic low = 0.8 volts.

Figure 56. Bus Arbitration Timing

Direct memory access

DIRECT MEMORY ACCESS INTERFACE (DMAI)

Originally published by Signetics February 1984

Preliminary

DESCRIPTION

The SCB68430 Direct Memory Access Interface (DMAI) is a single channel interface circuit which is intended to complement the performance and architectural capabilities of the SCN68000 microprocessor. The DMAI functions by transferring a series of operands (data) between memory and a device: operand sizes may be byte, word, or long word. A block is a sequence of operands: the number of operands in the block is determined by a transfer count stored within the DMAI. The SCN68430 can be programmed to utilize single cycle (cycle stealing) or burst data transfers.

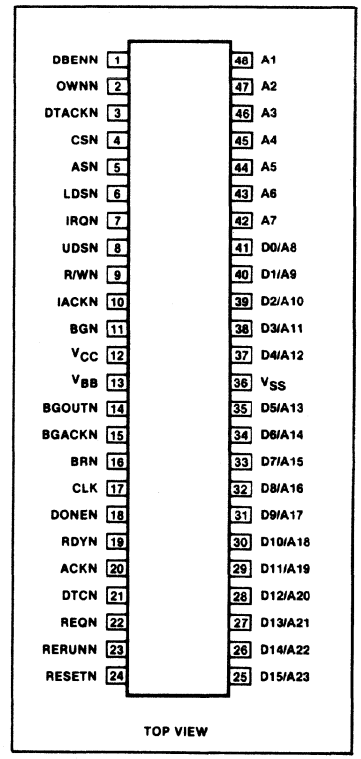
The DMAI provides two interfaces. The microprocessor interface is fully compatible with the SCN68000 microprocessor. The device interface includes lines for requesting, acknowledging, controlling, and timing the data transfers. The DMAI is a single-channel subset of the other 68000 family DMA controllers (68440 and 68450). It is software compatible with these devices and provides similar interfacing signals to both the system bus and the device.

The SCB68430 is constructed using Signetics ISL bipolar technology and is contained in a 48-pin dual-in-line package.

FEATURES

- **Bus compatible with SCN68000 microprocessor**
- **Software compatible with other 68K family DMA controllers**
- **Single address transfers**
- **Cycle steal and burst mode operation**
- **Bus arbitration daisy chain**
- **Automatic rerun on bus error**
- **Supports 32-bit transfers for VME bus**
- **Supports SCN68000 vectored interrupts**
- **24-bit address counter**
- **16-bit transfer counter**
- **Maximum transfer rate of 5 Mbytes per second**
- **Signetics ISL bipolar technology**

PIN CONFIGURATION

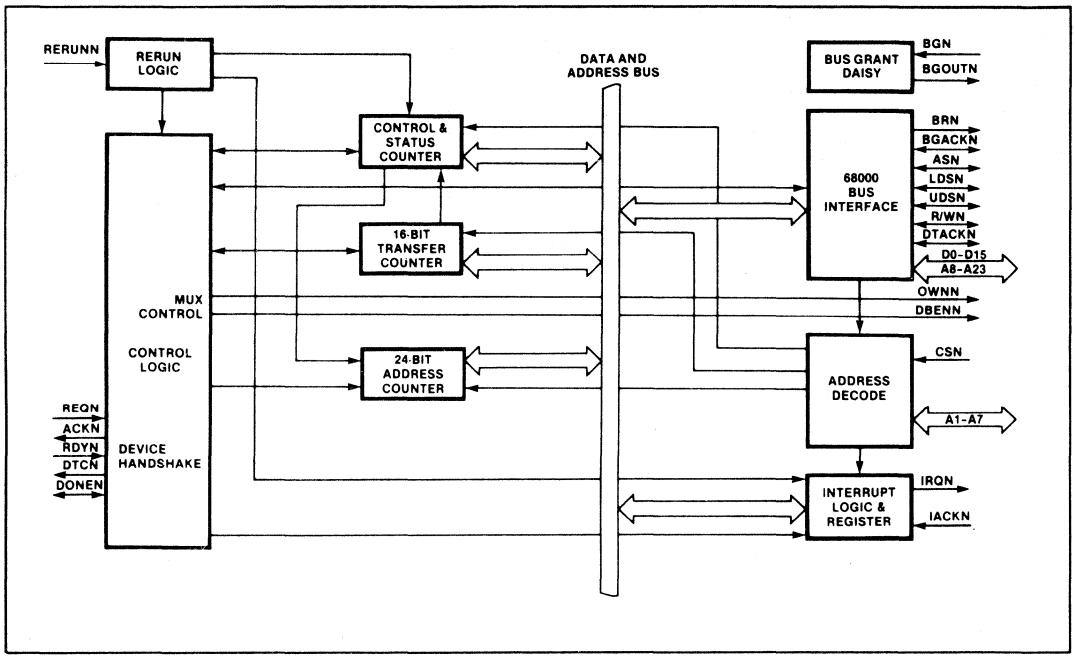


ORDERING CODE

Packages	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C	
	10MHz	12.5MHz
Ceramic DIP	SCB68430CAI48	SCB68430CCI48
Plastic DIP	SCB68430CAN48	SCB68430CCN48

Preliminary

BLOCK DIAGRAM



PIN DESIGNATIONS

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
A1-A7	48-42	I/O	**Address Lines:** Active high, three-state. In the MPU mode, these low order address lines specify which internal register of the DMAI is being accessed. In DMA mode, A1-A7 are outputs which provide the low order address bits of the location being accessed. Three-stated in IDLE Mode.
A8-A23/ D0-D15	41-37 35-25	I/O	**Address/Data Lines:** Active high, three-state. These lines are time multiplexed for data and address leads. The lines OWNN, RWN, CSN, and DBENN are used to control the demultiplexing of the address and data using external circuitry. In MPU mode, the bidirectional data lines (D0-D15) are used to transfer data between the MPU and the DMAI. In DMA mode, A8-A23 provide the high order address bits of the location being accessed. Three-stated in IDLE mode.
ASN	5	I/O	**Address Strobe:** Active low, three-state. In MPU and IDLE modes, ASN is an input which indicates that the current bus master has placed a valid address on the bus. It is monitored by the DMAI during bus arbitration to ascertain that the previous bus master has completed the current bus cycle. In DMA mode, it is an output indicating that the DMAI has placed a valid address on the bus.
UDSN	8	I/O	**Upper Data Strobe:** Active low, three-state. In MPU and IDLE modes, UDSN is an input which indicates that the upper data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
LDSN	6	I/O	**Lower Data Strobe:** Active low, three-state. In MPU and IDLE modes, LDSN is an input which indicates that the lower data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning.
R/WN	9	I/O	**Read/Write:** Active high for read, low for write, three-state. In MPU mode, R/WN is an input which controls the direction of data flow through the DMAI's input/output data bus interface and, if required, through an external data bus buffer. R/WN high causes the DMAI to place the data from the addressed register on the data bus, while R/WN low causes the DMAI to accept data from the data bus. In DMA mode, R/WN is an output to memory and I/O controllers indicating the type of bus cycle. It is held three-stated during IDLE mode.

Preliminary

PIN DESIGNATIONS (Continued)

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
CSN	4	I	**Chip Select:** Active low. When low, places the DMAI into the MPU mode. This input signal is used to select the DMAI for programmed data transfers. These transfers take place over D0-D15 as controlled by the R/WN and A1-A7 inputs. The DMAI is deselected when CSN is high. CSN is ignored during DMA mode.
DTACKN	3	I/O	**Data Transfer Acknowledge:** Active low, three-state. In MPU mode, DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate that valid data is present on the bus. The signal is negated (driven high) when completion of the cycle is indicated by negation of the CSN or IACKN input, and returns to the inactive third state a short time after it is negated. In DMA Mode, DTACKN is an input monitored by the DMAI to determine when the addressed device (memory) has latched the data (write cycle) or put valid data on the bus (read cycle).
RESETN	24	I	**Master Reset:** Active low. Assertion of this pin clears internal control registers (see table 1), initializes the interrupt vector register to H'0F', and sets the status register to the default value B'0000 000X', where X is the state of RDYN. All bidirectional I/O lines are three-stated and the DMAI is placed in the IDLE mode.
CLK	17	I	**Clock:** Active high. Usually the system clock, but may be any clock meeting the electrical specifications. Used by the DMAI to synchronize device functions and external control lines, and may not be gated off at any time.
IRQN	7	O	**Interrupt Request:** Active low, open collector. This output is asserted, if interrupts are enabled, upon end of transfer, on occurrence of a bus error, and on receipt of an abort from the MPU. The CPU can read the status register to determine the interrupting condition(s), or can respond with an interrupt acknowledge cycle to cause the DMAI to output an interrupt vector on the data bus.
IACKN	10	I	**Interrupt Acknowledge:** Active low. When asserted, indicates that the current cycle is an interrupt acknowledge cycle. The DMAI normally responds by placing the contents of the interrupt vector register on the data bus and asserting DTACKN. IACKN is not serviced if the DMAI has not generated an interrupt request.
BRN	16	O	**Bus Request:** Active low, open collector. BRN is asserted by the DMAI to request ownership of the bus after a DMA request is sensed on the REQN input from the I/O device. It is negated when the bus has been granted (BGN low) and BGACKN has been asserted, or, in burst DMA request mode, if the I/O device negates its request at least one clock cycle before BGACKN is asserted.
BGN	11	I	**Bus Grant:** Active low. BGN indicates to the DMAI that it is to be the next bus master. This signal is originated by the MPU and propagated via a daisy chain or other prioritization mechanism. After BGN is asserted, the DMAI waits until DTACKN, ASN, and BGACKN have become inactive before assuming ownership of the bus by asserting BGACKN.
BGOUTN	14	O	**Bus Grant Output:** Active low. Daisy chain output which is asserted by the DMAI when BGN is asserted and the DMAI does not have a bus request pending.
BGACKN	15	I/O	**Bus Grant Acknowledge:** Active low, three-state. As an input, BGACKN is monitored by the DMAI during the bus arbitration cycle to determine when it can assume ownership of the bus (BGACKN negated). In DMA mode, it is asserted by the DMAI to indicate that it is the bus master. Three-stated in MPU and IDLE modes.
RERUNN	23	I	**Rerun:** Active low. This input is asserted by external error detect logic to indicate a bus error. In DMA mode, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN. It remains halted until RERUNN becomes inactive, and then re-tries the last bus cycle. If RERUNN is asserted again, the DMAI sets the ERR bit in the status register, stops DMA operation, releases the bus, and interrupts the CPU, if interrupts are enabled, responding with a special interrupt vector when IACKN is asserted. Not monitored in MPU and IDLE modes.
REQN	22	I	**DMA Request:** Active low. This input from the I/O device requests service from the DMAI and causes the DMAI to request control of the bus. In burst mode, the input is level sensitive, and the DMAI releases the bus after REQN is negated and the current DMA cycle is completed. In cycle steal mode, the REQN input is negative edge triggered. A negative going edge must occur at least one clock cycle before DTCN is asserted to accomplish continuous transfer cycles.
ACKN	20	O	**DMA Request Acknowledge:** Active low. ACKN is asserted by the DMAI to indicate that it has gained the bus and the requested bus cycle is now beginning. It is asserted at the beginning of every bus cycle after ASN has been asserted, and is negated at the end of every bus cycle.

Preliminary

PIN DESIGNATIONS (Continued)

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
RDYN	19	I	**Device Ready:** Active low. RDYN is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states. RDYN can be held low continuously if the device is fast enough so that wait states are not required.
DTCN	21	O	**Device Transfer Complete:** Active low. In DMA mode, DTCN is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched.
DONEN	18	I/O	**Done:** Active low, open collector. As an output, DONEN is asserted by the DMAI concurrent with the ACKN output to indicate to the device that the transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer. As an input, if asserted by the device before the transfer count became zero, it causes the DMAI to abort service and generate an interrupt request, if interrupts are enabled.
OWNN	2	O	**Own:** Active low, open collector. This output is asserted by DMAI during the DMA mode to indicate bus mastership. It can be used to enable external address/data and control buffers. Inactive in MPU and IDLE modes.
DBENN	1	O	**Data Bus Enable:** Active low, open collector. Asserted by the DMAI when CSN is asserted or when IACKN is asserted and the DMAI has an interrupt request pending. Can be used to enable bidirectional data buffers for D0-D15. Inactive in IDLE mode.
V_{CC}	12	I	**Power Supply:** + 5 volt power input.
V_{BB}	13	I	**Power Supply:** + 1.5 volt power input.
V_{SS}	36	I	**Ground:** Signal and power ground input.

PIN DESCRIPTION

The Pin Designation table describes the function of each of the pins of the DMAI. Signal names ending in 'N' are active low. All other signals are active high. In the descriptions, 'MPU mode' refers to the state when the DMAI is chip selected. The term 'DMA mode' refers to the state when the DMAI assumes ownership of the bus. The DMAI is in the 'IDLE mode' at all other times.

In this data sheet signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic zero) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

REGISTERS AND COUNTERS

Register Map

The internal accessible register organization of the DMAI is shown in table 1. The following rules apply to all registers:

1. A read from a reserved location in the map results in a read from the the 'null register'. The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle but no write occurs.
2. Unused bits of a defined register are read as indicated in the register descriptions.
3. All registers are addressable as 8-bit quantities. To facilitate operation with the 68K MOVEP instruction, addresses are ordered such that certain sets of registers may also be accessed as words or long words.

The operation of the DMAI is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The contents of certain control and status registers are initialized on RESET.

To provide compatibility with the other 68K family DMA controllers, control and status bits are mapped in bit positions equivalent to where they are located in the register map of the other devices. Bits which are used in the other devices but not in the DMAI are assigned default values. If upward compatibility to the other controllers is required, the programmer should use these default values when writing the control words to the registers, although they have no effect in the DMAI. When a register is read, the default value is returned regardless of the value used when the register is programmed. The default value is indicated by '(x)' in unused bit positions in the register formats, which are illustrated in table 2.

Device Control Register (DCR)

[15] External Request Mode. This bit selects whether the DMAI operates in burst or cycle steal mode.

0 Burst mode. This mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request (REQN) line is an active low input which is asserted by the device to request an operand transfer. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request.

1 Cycle steal mode. In this mode, the device requests an operand transfer by generating a falling edge on the request (REQN) line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and

Preliminary

notifying the peripheral by asserting the acknowledge (ACKN) output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Operation Control Register (OCR)

[7] Direction

0 Transfer is from memory to device.

1 Transfer is from device to memory.

[5:4] Operand Size. The programming of these bits determine whether UDSN, LDSN, or both are generated during the transfer cycle and the increment by which the memory address counter (MAC) is changed in each transfer cycle.

00 Byte. The operand size is 8 bits. The MAC is incremented by one after each operand transfer. If the LSB of the MAC is a '0', UDSN is asserted during the transfer. If the LSB of a MAC is a '1', LDSN is asserted during the transfer. The transfer counter decrements by one before each byte is transferred.

01 Word. the operand size is 16 bits. The MAC is incremented by two after each operand transfer. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before each word is transferred.

10 Long word. The operand size is 32 bits. The operand is transferred as two 16-bit words. The MAC is incremented by two after each 16-bit word is transferred. The value of the LSB of the MAC is ignored and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the entire long word is transferred. Note that this mode is not implemented in the 68440.

11 Double word. The operand size is 32 bits. The operand is transferred as a single 32-bit word. The MAC is incremented by four after each operand transfer. The value of the two LSBs of the MAC is ignored (the A1 output will always be a zero in this mode) and both UDSN and LDSN are asserted during the transfer. The transfer counter decrements by one before the double word is transferred. Note that this mode is not implemented in the 68440 or 68450; it is included in the DMAI to support VME bus operations.

Table 1. DMAI ADDRESS MAP

ADDRESS BITS[1,2]								ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
7	6	5	4	3	2	1	0				
d	d	0	0	0	0	0	0	CSR	Channel Status Register	R/W[3]	Yes
d	d	0	0	0	0	0	1	CER	Channel Error Register	R	Yes
d	d	0	0	0	0	1	0		Reserved		
d	d	0	0	0	0	1	1		Reserved		
d	d	0	0	0	1	0	0	DCR	Device Control Register	R/W	Yes
d	d	0	0	0	1	0	1	OCR	Operation Control Register	R/W	Yes
d	d	0	0	0	1	1	0	SCR	Sequence Control Register	R/W[4]	Yes
d	d	0	0	0	1	1	1	CCR	Channel Control Register	R/W	Yes
d	d	0	0	1	0	0	0		Reserved		
d	d	0	0	1	0	0	1		Reserved		
d	d	0	0	1	0	1	0	MTCH	Memory Transfer Counter High	R/W	No
d	d	0	0	1	0	1	1	MTCL	Memory Transfer Counter Low	R/W	No
d	d	0	0	1	1	0	0	MACH	Memory Address Counter High	R/W[4]	No
d	d	0	0	1	1	0	1	MACMH	Memory Address Counter Middle High	R/W	No
d	d	0	0	1	1	1	0	MACML	Memory Address Counter Middle Low	R/W	No
d	d	0	0	1	1	1	1	MACL	Memory Address Counter Low	R/W	No
d	d	0	1	d	d	d	d		Reserved		
d	d	1	0	0	0	d	d		Reserved		
d	d	1	0	0	1	0	0		Reserved		
d	d	1	0	0	1	0	1	IVR	Interrupt Vector Register	R/W	Yes
d	d	1	0	0	1	1	0		Reserved		
d	d	1	0	0	1	1	1	IVR	Interrupt Vector Register	R/W	Yes
d	d	1	0	1	0	d	d		Reserved		
d	d	1	0	1	1	0	0		Reserved		
d	d	1	0	1	1	0	1	CPR	Channel Priority Register	R/W[4]	No
d	d	1	0	1	1	1	0		Reserved		
d	d	1	0	1	1	1	1		Reserved		
d	d	1	1	d	d	d	d		Reserved		

Notes:
1. A0 = 0 for UDSN asserted, A0 = 1 for LDSN asserted.
2. 'd' designates don't care.
3. A write to this register may perform a status resetting operation.
4. This register is a dummy register present only to provide compatibility with other 68K family DMA controllers. A write to this register has no effect on the DMAI.

Preliminary

Table 2. REGISTER BIT FORMATS

DEVICE CONTROL REGISTER

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
DCR	EXTERNAL REQUEST MODE 0 = BURST 1 = CYCLE STEAL	NOT USED (0)	NOT USED (1)	NOT USED (1)	NOT USED (*)	NOT USED (0)	NOT USED (0)	NOT USED (0)

*Should be programmed as '0' for SIZE (OCR[5:4]) = 00 and as '1' otherwise. When read, the value of this bit is OCR[5].OR.OCR[4].

OPERATION CONTROL REGISTER (OCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
OCR	DIRECTION 0 = MEM TO DEV 1 = DEV TO MEM	NOT USED (0)	OPERAND SIZE 00 = BYTE 01 = WORD (16 BIT) 10 = LONG WORD 11 = WORD (32-BIT)		NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)

*Long word and 32-bit word modes are not supported by 68440. 32-bit word mode is not supported by 68450.

SEQUENCE CONTROL REGISTER (SCR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
SCR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (1)	NOT USED (0)	NOT USED (0)

CHANNEL CONTROL REGISTER (CCR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CCR	START 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	SOFTWARE ABORT 0 = NO 1 = YES	INTERRUPT ENABLE 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	NOT USED (0)

CHANNEL STATUS REGISTER (CSR)

	BIT15	BIT14	BIT13	BIT12	BIT11	BIT10	BIT09	BIT08
CSR	CHANNEL OPERATION COMPLETE 0 = NO 1 = YES	NOT USED (0)	NORMAL DEVICE TERMINATE 0 = NO 1 = YES	ERROR 0 = NO 1 = YES	CHANNEL ACTIVE 0 = NO 1 = YES	NOT USED (0)	NOT USED (0)	READY INPUT STATE 0 = LOW 1 = HIGH

CHANNEL ERROR REGISTER (CER)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CER	NOT USED (0)	NOT USED (0)	NOT USED (0)	ERROR CODE 00000 = NO ERROR 01001 = BUS ERROR 10001 = SOFTWARE ABORT				

CHANNEL PRIORITY REGISTER (CPR)

	BIT07	BIT06	BIT05	BIT04	BIT03	BIT02	BIT01	BIT00
CPR	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)	NOT USED (0)

Preliminary

Sequence Control Register (SCR)

This register serves no function in the DMAI. It is included only to provide compatibility with the programming for the 68440 and 68450 DMA controllers.

Channel Control Register (CCR)

[7] Start Operation

0 No start pending.

1 Start operation. The start bit is set to initiate operation of the DMAI. The memory address counter and the memory transfer counter should have been previously initialized, and all bits of the channel status register (CSR) should have previously been reset. The DMAI initiates operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive requests for an operation. The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared.

A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

[4] Software Abort

0 Do not abort.

1 Abort operation. Setting this bit terminates the current operation of the DMAI and places it in the IDLE state. The channel operation complete and error bits in the CSR are set, the channel active bit in the CSR is reset, and an ABORT ERROR condition is signaled in the CER. Setting this bit causes a pending start to be reset.

[3] Interrupt Enable

0 Interrupts not enabled.

1 Enable interrupts. An interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, the DMAI returns the normal interrupt vector if the error bit in the CSR is not set, or the error interrupt vector if error is set.

Channel Status Register (CSR)

A read of this register provides the status of the DMAI. The COC, NDT, and ERR bits can be cleared by writing a '1' to the bit positions of the register which are to be cleared. Those bit positions which are written with a '0' remain unaffected.

[15] Channel Operation Complete. This bit is set following the termination, whether successful or not, of any DMAI operation and indicates that the DMA transfer has completed. This bit must be cleared to start another channel operation.

[13] Normal Device Termination. This bit is set when the device terminates the DMAI operation by asserting the DONEN line while the device was being acknowledged. This bit must be cleared to start another channel operation.

[12] Error. This bit is used to report that the DMAI's operation was terminated due to the occurrence of an error. The condition which caused the error can be determined by reading the channel error register (CER). This bit must be cleared to start another channel operation. When this bit is cleared, the CER is also cleared.

[11] Channel Active. This bit is set after the channel has been started and remains set until the channel operation terminates. It is then automatically cleared by the DMAI. The bit is unaffected by the write operations.

[8] Ready Input State. This bit reflects the state of the RDYN input at the time the CSR is read. The bit is a '0' if RDYN is low and a '1' if RDYN is high. This bit is unaffected by write or reset operations.

Channel Error Register (CER)

[4:0] Error Code. This field indicates the source of error when an error is indicated in CER[12]. The contents of this register are cleared when CER[12] is cleared.

00000 No error.

01001 Bus error. A bus error occurred during the last bus cycle generated by the DMAI. See rerun description in OPERATION section.

10001 Software abort. The channel operation was terminated by a software abort command. See CCR[4].

Channel Priority Register (CPR)

This register serves no function in the DMAI. It is included only to provide compatiblity with the programming for the other 68K family DMA controllers.

Memory Address Counter (MACH, MACMH, MACML, MACL)

The 32-bit memory address counter is used to program the memory location where the first operand to be transferred is located or is to be transferred to, depending on the direction of transfer. The counter must be initialized prior to beginning the transfer of a block of data and then increments automatically depending on the operand length, as described in the Operation Control Register description.

Only the least significant 24 bits of the counter (MACMH, MACML, and MACL) are implemented in the DMAI. The most significant byte of the counter, MACH, is provided only to allow compatibility with programming of the 68440 and 68450. Writing to MACH has no effect on the DMAI operation. Reading MACH always returns H'00'.

Memory Transfer Counter (MTCH, MTCL)

The 16-bit memory transfer counter programs the number of operands to be transferred by the DMAI. The counter must be initialized prior to beginning the transfer of a block of data and then decrements once per operand transfer (regardless of operand size) until it reaches the terminal value of zero. Channel operation then terminates and the COC bit in the CSR will be asserted.

Interrupt Vector Register (IVR)

The IVR contains the value to be placed on the data bus upon receipt of an interrupt acknowledge from the MPU. Only the seven most significant bits of the programmed value are used by the DMAI. The output vector from the DMAI contains a zero in the least significant bit position if a normal termination occurred (error bit not set) and contains a one in the least significant bit position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

OPERATION

A DMAI operation proceeds in three principal phases. During the initialization phase, the MPU configures the channel control registers, loads the initial memory address and transfer count, and starts the channel. During the transfer phase, the DMAI accepts requests for transfers from the device, arbitrates for and acquires ownership of the bus, and provides for addressing and bus controls for the transfers. The termination phase occurs after the operation is complete, when the DMAI reports the status of the operation.

Preliminary

Operation Initiation

After having programmed the control registers, the memory address counter, and the memory transfer counter, the MPU sets the start bit (CCR[7]). The DMAI initiates the operation by clearing any pending requests, clearing the start bit, and setting the channel active bit in the CSR. The DMAI is then ready to receive valid requests for an operation.

The channel cannot be started if any of the internal status bits in the CSR (CSR[15:11]) have not been cleared. An error is not signaled if this condition occurs. The only indication of this state is that the start bit remains set in the CCR. A pending start cannot be reset by a write to the register. START can be cleared only by the DMAI when it starts operation or by setting the software abort bit (CCR[4]).

Device/DMAI Communication

Communication between the peripheral device and the DMAI is accommodated by five signal lines:

Request (REQN). The device makes a request for service by asserting the request line. The DMAI can operate in either the burst request mode or the cycle stealing request mode, as programmed by the external request mode bit (DCR[15]).

The burst mode allows a device to request the transfer of multiple operands using consecutive bus cycles. In this mode the request line is an active low input. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. If the request line is active when the DMAI asserts ACKN, and remains active at least until the DMAI asserts device transfer complete (DTCN), the DMAI recognizes a valid request for another operand, which will be transferred during the next bus cycle. If the request line is negated before the DMAI asserts DTCN, the DMAI relinquishes the bus and waits for the next request. For long word transfers (2 x 16), the request must be asserted at least until the acknowledge for the second part of the operand has been asserted.

In the cycle steal mode, the device requests an operand transfer by generating a falling edge on the request line. The DMAI services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge (ACKN) output. The request line must be in the inactive state for at least one clock cycle before a request is made. After a request has been asserted, it must remain at the assertion level for at least one clock cycle. If another request is received before the first operand part of a former request is acknowledged, the second request is not recognized. Normally, the DMAI will relinquish the bus after servicing a valid request. However, if the device generates a new request before the DMAI asserts DTCN for the last operand part, the DMAI will retain ownership of the bus and that request will be serviced before the bus is relinquished.

Acknowledge (ACKN). The DMAI asserts the acknowledge line, which implicitly addresses the device making the request, during transfers to and from the device. The line may be used to control buffering circuits between the data bus and the MPU bus.

During burst mode, REQN must not be disasserted for less than one CLK period plus four RC time constants, where R is the value of the resistor used for the pullup on BRN and C has a typical value of 20pF.

Ready (RDYN). Ready is an active low input which is asserted by the requesting device to indicate to the DMAI that valid data has either been stored or put on the bus. If negated, it indicates that the data has not been stored or presented, causing the DMAI to enter wait states until RDYN is asserted. RDYN can be held low continuously if the device is fast enough so that wait states are not required. The current state of the ready input is reflected in CSR[8].

Done (DONEN). Done is a bidirectional active low signal. As an output, it is asserted and negated by the DMAI concurrent with the ACKN output of the last operand part to indicate to the device that the memory transfer count is exhausted and that the DMAI's operation is completed as a result of that transfer.

The DMAI also monitors the state of the line while acknowledging a device. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. In this case the DMAI clears the channel active bit and sets the channel operation complete and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Device Transfer Complete (DTCN). DTCN is an active low output which is asserted by the DMAI to indicate to the device that the requested data transfer is complete. On a write to memory operation, it indicates that the data provided by the device has been successfully stored. On a read from memory operation, it indicates to the device that the data from memory is present on the data bus and should be latched. DTCN is not asserted if assertion of the RERUNN input terminates the bus cycle.

Bus Arbitration

Upon receiving a valid request for a transfer from the device, the DMAI will arbitrate for and obtain ownership of the system bus.

The DMAI indicates that it wishes to become the bus master by asserting its bus request (BRN) output. This is a wire-ORed signal that indicates to the MPU that some external device requires control of the bus. The processor is effectively at a lower priority level than external devices and will relinquish the bus after it has completed the last bus cycle it has started. The processor puts the bus up for external arbitration by asserting its bus grant (BGN) output. This signal may be routed through a daisy chain (such as provided by the DMAI) or through some other priority-encoded network. When the DMAI making the bus request receives the bus grant (indicated by its BGN input being asserted), it is to be the next bus master. It waits until address strobe (ASN), data transfer acknowledge (DTACKN) and bus grant acknowledge (BGACKN) become inactive and then assumes ownership of the bus by asserting its own BGACKN output. The DMAI then negates the BRN output and proceeds with the data transfer phase. After this phase is completed, the DMAI relinquishes bus ownership by negating the BGACKN output.

In burst DMA mode, detection of an active low request input after the DMAI operation has been started will begin the bus arbitration cycle. However, if the device negates its request at least one clock cycle before the DMAI asserts BGACKN, the DMAI will negate its bus request and will not assume ownership of the bus.

Data Transfers

The actual transfer of data between the memory and the device occurs during the data transfer phase. All transfers occur during a single cycle except in the case of long word operands, in which case two cycles are used to transfer the operand as two 16-bit words. The transfers take place using a 'single address' protocol; the DMAI addresses the memory via the bus address lines, while the device is implicitly addressed via the acknowledge output.

When a request is generated using the request method programmed in the control register, the DMAI obtains the bus and

Preliminary

asserts acknowledge to notify the device that a transfer is to take place. The DMAI asserts all S68000 bus control signals needed for the transfer and holds them until the device responds with ready. The bus cycle then terminates normally. Ready may be tied low (asserted) if the device is fast enough.

When the transfer is from memory to the device, data is valid when DTACKN is asserted by the memory and remains valid until the data strobe(s) are negated. The assertion of DTCN from the DMAI can be used to latch the data, as the data strobes are not removed until one-half clock after the assertion of DTCN.

When the transfer is from device to memory, the data must be valid on the bus before the DMAI asserts the data strobe(s). The device indicates valid data by asserting ready. The DMAI then asserts the strobes and holds them asserted until the memory accepts the data, indicated by the assertion of DTACKN. The DMAI then asserts DTCN and negates the data strobes.

Flow charts for these operations are shown in figures 1 and 2. Refer to the timing section for the equivalent timing diagrams.

Operation Termination

Termination of the block transfer occurs under the conditions detailed below.

Terminal Count. As part of each transfer of an operand, the DMAI decrements the memory transfer counter. If this counter is decremented to zero, the operand is the last operand of the block. The DMAI operation is complete and it notifies the device of completion by asserting the DONEN output during the last operand transfer cycle. When the transfer has been completed, the channel active bit in the CSR is cleared and the COC bit is set.

Device Termination. The DMAI monitors the state of the DONEN line while acknowledging a device transfer request. If the device asserts DONEN, the DMAI will terminate operation after the transfer of the current operand. When the transfer has been completed, the DMAI clears the channel active bit and sets the COC and normal device termination bits in the CSR. If both the DMAI and the device assert DONEN, the device termination is not recognized, but the operation does terminate.

Software Abort. The software abort bit (CCR[4]) allows the MPU to abort the current operation of the DMAI. The COC and error bits in the CSR are set, the channel active bit in the CSR is cleared, and an abort error condition is signaled in the CER.

Rerun Error. The DMAI provides a rerun input (RERUNN) to indicate a bus exception condition. RERUNN must arrive prior to or in coincidence with DTACKN in order to be recognized, and the DMAI verifies that the line has been stable for two clock cycles before acting on it. The occurrence of a rerun during a DMAI bus cycle forces it to terminate the bus cycle in an orderly manner.

When the assertion of rerun is verified, the DMAI stops operation and three-states the data, address, and control lines, except BGACKN, so that it retains ownership of the bus. It remains halted until rerun becomes inactive, and then re-tries the last bus cycle. If rerun is asserted again, the DMAI stops DMA operation, releases the bus, sets the error and COC bits in the CSR, clears the active bit in the CSR, and sets the error code in the CER to indicate a bus error.

While stopped due to assertion of rerun, the DMAI does not generate any bus cycles and will not honor any requests until it is removed. However, the DMAI still recognizes requests.

Error Recovery Procedure. If an error occurs during a DMA transfer such that the DMAI stops the DMA operation, information is available to the operating system for an error recovery routine.

The information available to the operating system consists of the memory address counter, the memory transfer counter, and the control, status, and error registers. The DMAI decrements the memory transfer counter before attempting a DMA operation, so the register will contain the count minus one of the attempted transfer. The memory address counter will contain the address at which the DMA operation was attempted.

Reset. The reset input (RESETN) provides a means of resetting and initializing the DMAI from an external source. If the DMAI is a bus master when reset is received, the DMAI relinquishes the bus. Reset clears the control and error registers, sets all bits of the status register except CSR[8] to zero, and initializes the interrupt vector register to H'0F'.

Interrupts. The interrupt enable bit (CCR[3]) determines whether the DMAI generates interrupt requests. When the bit is set, an interrupt request is generated if the channel operation complete bit in the CSR is set. When the IACKN input is asserted, and the DMAI has an interrupt request pending, the DMAI returns an interrupt vector on the data bus.

The interrupt vector issued is the contents of the IVR. Only the seven most significant bits of the programmed value are used by the DMAI. The vector from the DMAI contains a zero in the LSB position if a normal termination occurred (error bit not set) and contains a one in the LSB position if termination was due to an error (error bit set).

The contents of this register are initialized to H'0F' by a reset. The value returned will be H'0F', regardless of the error state, until the register is programmed by the MPU.

To provide compatibility with the other 68K family DMA controllers, the IVR has two addresses (see table 1). If program compatibility is required, the value written at the normal IVR address should have a zero as its LSB, and the value written at the error IVR address should be the same but with the LSB equal to one.

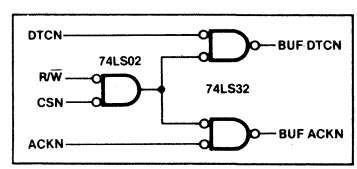
APPLICATIONS

Figure 3 illustrates a typical interconnection of the DMAI in a 68000 based system.

DESIGN NOTE

When clearing the error bit in CSR (bit 12) after a DMAI abort due to a double RERUNN, ACKN and DTCN will both go low concurrent with CSN and DTACKN for one CLK cycle.

To prevent the possibility that the device may mis-interpret these signals, it is suggested that these signals be ANDed with CSN (see figure below).



Preliminary

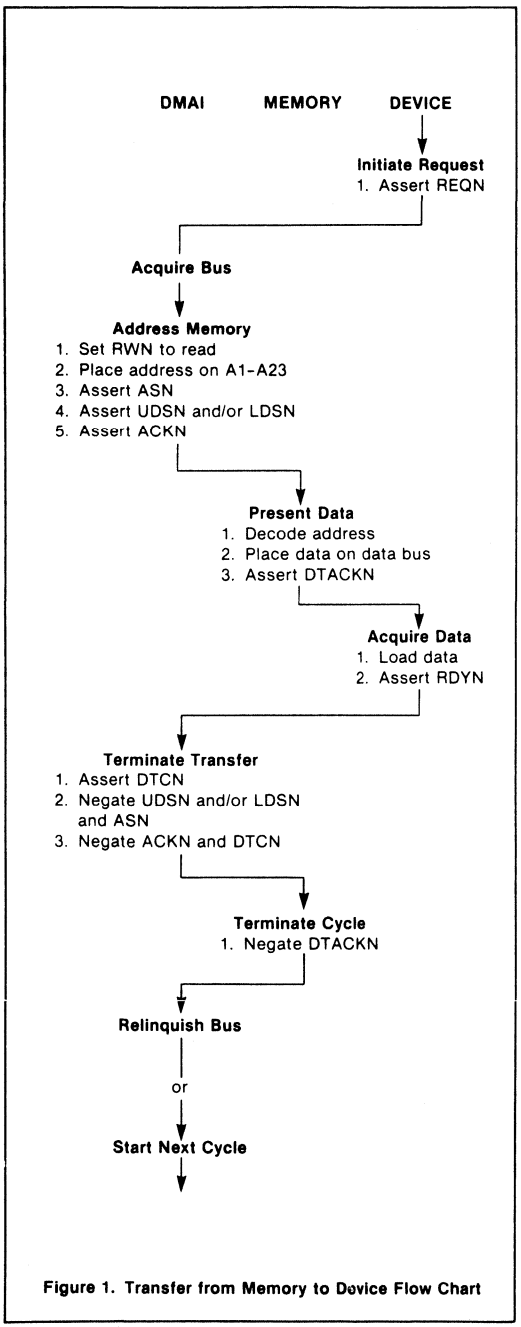


Figure 1. Transfer from Memory to Device Flow Chart

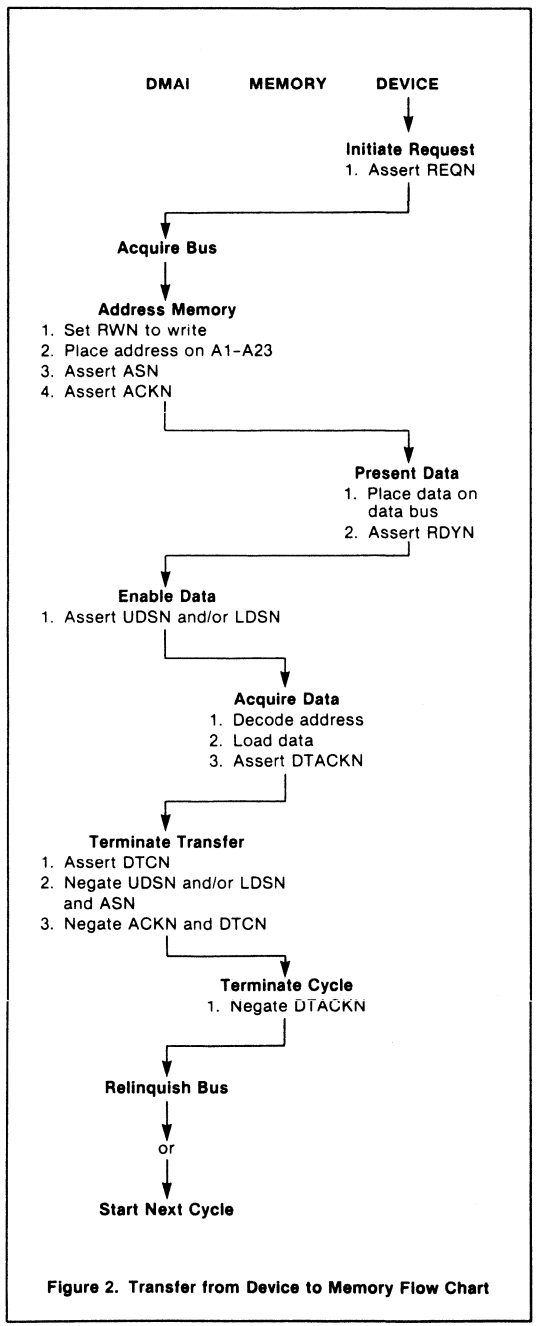


Figure 2. Transfer from Device to Memory Flow Chart

Preliminary

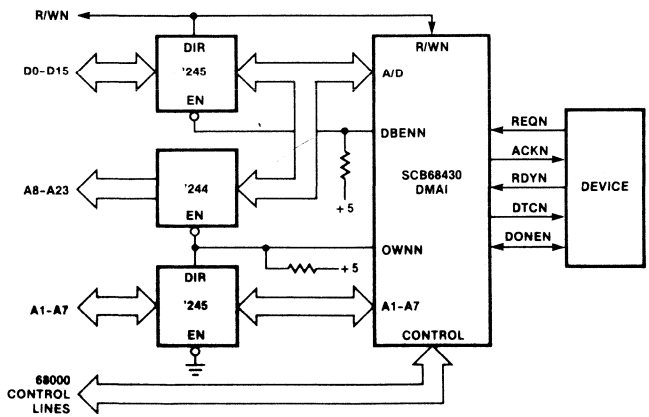


Figure 3. DMAI Application

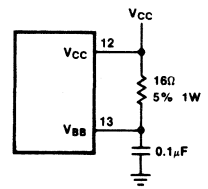


Figure 4. Recommended V_{BB} Test Circuit

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltages V_{CC} and V_{BB}	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V ±5%, V_{BB} = Figure 4, T_A = 0°C to +70°C[3,4,7]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Max	UNIT
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
V_{OL}	Output low voltage	I_{OUT} = 5.3mA		0.5	V
V_{OH}	Output high voltage, all outputs except open collector outputs[5]	I_{OUT} = −400μA	2.5		V
I_{IL}	Input low current	V_{IN} = 0.4V		−400	μA
I_{IH}	Input high current	V_{IN} = 2.7V		20	μA
I_{OC}	Open collector off state current[5]	V_{OUT} = 2.4V		20	μA
I_{SC}	Output short circuit current[6]	V_{CC} = max	−40	−100	mA
I_{CC}	V_{CC} supply current	V_{CC} = max, V_{BB} = max		130	mA
I_{BB}	V_{BB} supply current			275	mA

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.
5. IRQN, BRN, DONEN, and OWNN are open collector outputs.
6. No more than one output should be connected to ground at one time.
7. Capacitive test load is 100pF for all pins except DTCN which has a 35pF capacitive test load.

Preliminary

AC ELECTRICAL CHARACTERISTICS $V_{CC} = 5.0V \pm 5\%$, V_{BB} = Figure 4, $T_A = 0°C$ to $+70°C$[3,4,7]

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
1	5	A1-A7, ASN, RWN setup to UDSN, LDSN low	0		0		ns
2	5	D0-D15 3-state to invalid data from ASN, CSN, and UDSN or LDSN low	10		10		ns
3	5	DTACKN 3-state to high from ASN, CSN, and UDSN or LDSN low	10		10		ns
4	5	CSN low after UDSN or LDSN low		25		25	ns
5	5, 6	DBENN low after ASN and CSN low		60		60	ns
6	5	D0-D15 valid data from ASN, CSN, and UDSN or LDSN low		100		100	ns
7	5	DTACKN low after D0-D15 valid data	−15	30	−15	30	ns
8	5	A1-A7, ASN, RWN or CSN hold after UDSN and LDSN high	0		0		ns
9	5, 6	DBENN high from either ASN or CSN high		45		45	ns
10	5	D0-D15 to 3-state from UDSN and LDSN high		80		80	ns
11	5	D0-D15 to invalid data from UDSN and LDSN high	10		10		ns
12	5, 6	DTACKN high from UDSN and LDSN high		55		55	ns
13	5, 6	DTACK 3-state from either CSN or ASN high		85		85	ns
14	6	A1-A7, ASN, RWN setup to UDSN, LDSN low	50		50		ns
15	6	CSN setup before UDSN or LDSN low	20		20		ns
16	6	DTACKN 3-state to high after CSN and ASN low	10		10		ns
17	6	D0-D15 valid after UDSN or LDSN low		0		0	ns
18	6	DTACKN low from UDSN or LDSN low		100		100	ns
19	6	UDSN and LDSN low time	115		100		ns
20	6	A1-A7 hold after UDSN and LDSN high	0		0		ns
21	6	ASN, RWN and CSN hold after UDSN and LDSN high	0		0		ns
22	6	D0-D15 hold after UDSN and LDSN high	0		0		ns
23	7	DBENN low from last low of ASN, IACKN, LDSN		65		65	ns
24	7	D0-D7 valid after last low of ASN, IACKN, LDSN		105		105	ns
25	7	DTACKN 3-state to high after last low of ASN, IACKN, LDSN		100		100	ns
26	7	DTACKN low after last low of ASN, IACKN, LDSN		110		110	ns
27	7	DBENN high after first high of ASN, IACKN, LDSN		55		55	ns
28	7	D0-D7 hold after first high of ASN, IACKN, LDSN		60		60	ns
29	7	D0-D7 3-state after first high of ASN, IACKN, LDSN		80		80	ns
30	7	DTACKN high after first high of ASN, IACKN, LDSN		60		60	ns
31	7	DTACKN 3-state after first high of ASN, IACKN, LDSN		95		95	ns
32	8	BRN high from CLK high		65		65	ns
33	8, 11, 12	BGACKN low from CLK low		75		75	ns
34	8, 11, 12	OWNN low from CLK high		75		75	ns
35	8	BGACKN high from CLK low		75		75	ns
36	8, 11, 12	OWNN high from CLK high (load dependent)		50		50	ns
37	10	REQN setup before CLK low	30		30		ns
38	10	REQN hold after CLK high	20		20		ns
39	10	BRN low from CLK high		80		80	ns
41	11, 12	ASN, UDSN, LDSN, RWN 3-state to high from CLK low		75		75	ns
43	11, 12	A1-A23 to valid ASN	0		0		ns
44	11, 12	ASN low from CLK high		65		65	ns
45	11, 12	LDSN, UDSN low from CLK high		90		90	ns
46	11, 12	ACKN low from CLK high		65		65	ns
47	11, 12	DTACKN setup to CLK high	30		30		ns
48	11, 12	RDYN setup to CLK low	30		30		ns
49	11, 12	DTCN low from CLK high		70		70	ns
50	11, 12	ASN high from CLK high		75		75	ns
51	11, 12	LDSN, UDSN high from CLK high		90		90	ns
52	11, 12	DTACKN, RDYN hold after CLK high	0		0		ns
—	11, 12	ASN, LDSN, UDSN high from DTCN low	−20		−20		ns
53	11, 12	ACKN high from CLK high		50		50	ns
54	11, 12	DTCN high from CLK high		50		50	ns
55	11, 12	Address valid after CLK low	10		10		ns
—	11, 12	Address valid after ASN high	0		10		ns
56	11, 12	DONEN (output) low from CLK low		120		120	ns

Preliminary

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS				UNIT
			10MHz		12.5MHz		
			Min	Max	Min	Max	
57	11, 12	DONEN (output) high from CLK high		50		50	ns
58	11, 12	DONEN (input) setup low before CLK low	30		30		ns
59	11, 12	DONEN (input) hold low after CLK high	0		0		ns
60	11, 12	BGACKN, ASN, UDSN, LDSN, RWN to 3-state from CLK low		75		75	ns
62	11, 12	A1-A23 valid to 3-state from CLK high		100		100	ns
63	12	R/WN low from CLK high		65		65	ns
64	12	R/WN high from CLK high		75		75	ns
65	13	RERUNN setup low before CLK high	30		30		ns
66	13	RERUNN hold low from CLK high	20		20		ns
67	13	A1-A23 to idle state from CLK low		100		100	ns
68	13	A1-A23 to valid after CLK low		85		85	ns

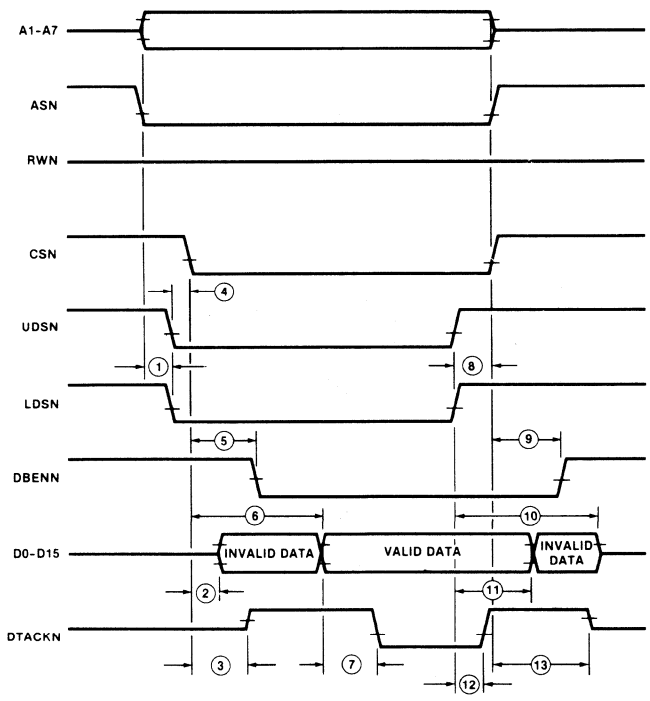


Figure 5. DMAI Read Timing

Preliminary

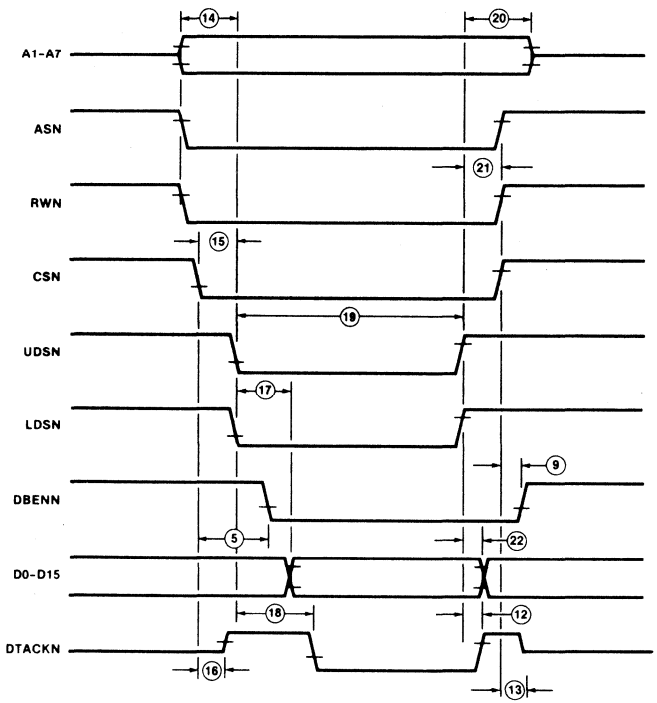


Figure 6. DMAI Write Timing

Preliminary

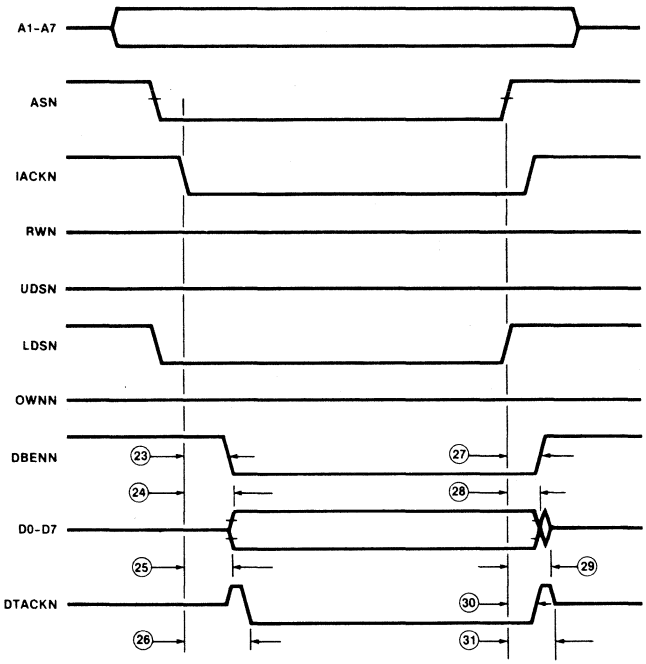


Figure 7. CPU IACK Cycle to DMAI

Preliminary

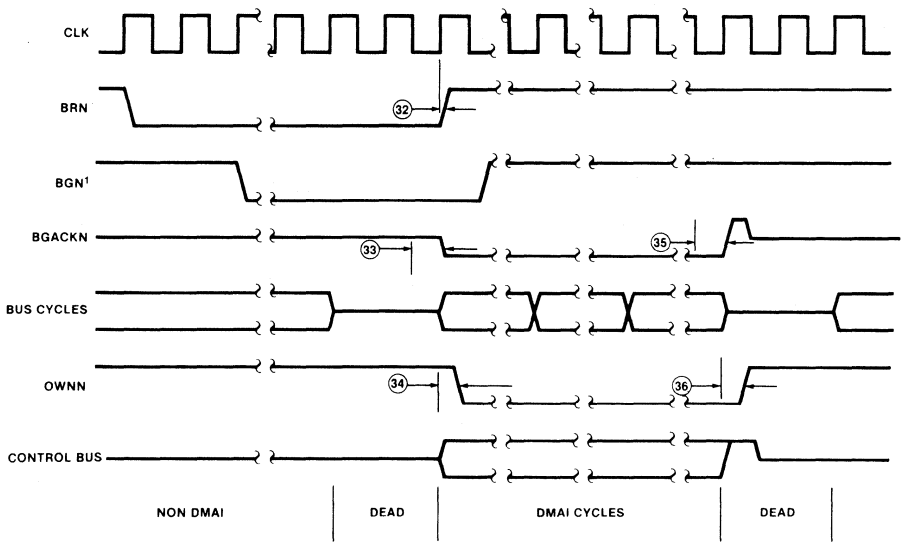
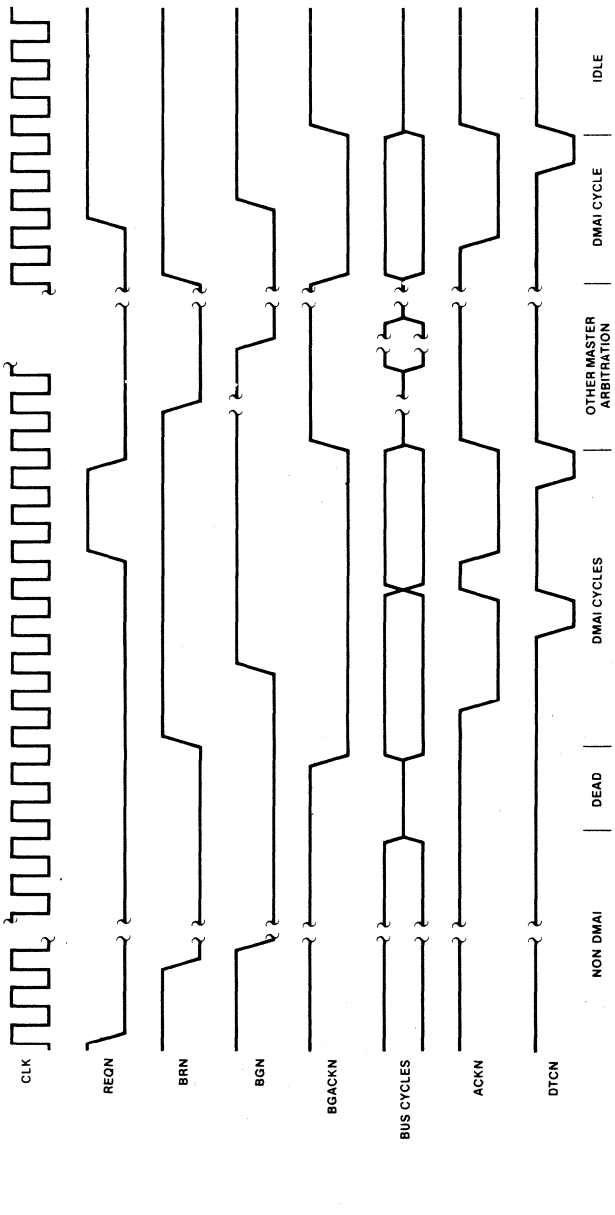


Figure 8. DMAI Bus Arbitration Timing

NOTES:
1. Device will become master if BGN is asserted concurrent with or later than REQN (same clock edge or later).
2. ASN, DTACKN and BGACKN must be negated in order for DMAI to become master. Timing assumes all these happen concurrent with BGN — if not, it is from the latest signal which is negated.

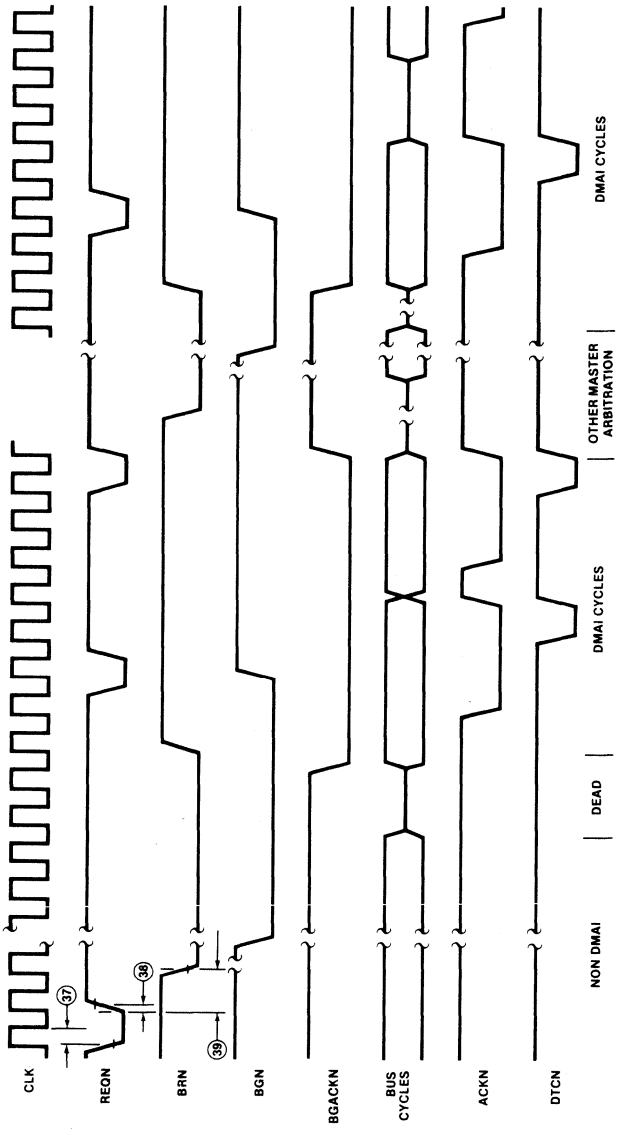
Preliminary



NOTES:
1. To maintain mastership next bus cycle, REQN must remain asserted at least until 1/2 clock cycle after DTCN is asserted.
2. To guarantee that mastership is relinquished next cycle, REQN must be negated no later than 1/2 clock cycle prior to DTCN.

Figure 9. DMAI Burst Mode Request Timing

Preliminary



NOTE:
1. In order to keep the bus, REQN must come no later than the 1/2 clock minus the setup time (37) prior to assertion edge of DTCN.

Figure 10. DMAI Cycle Steal Mode Request Timing

Preliminary

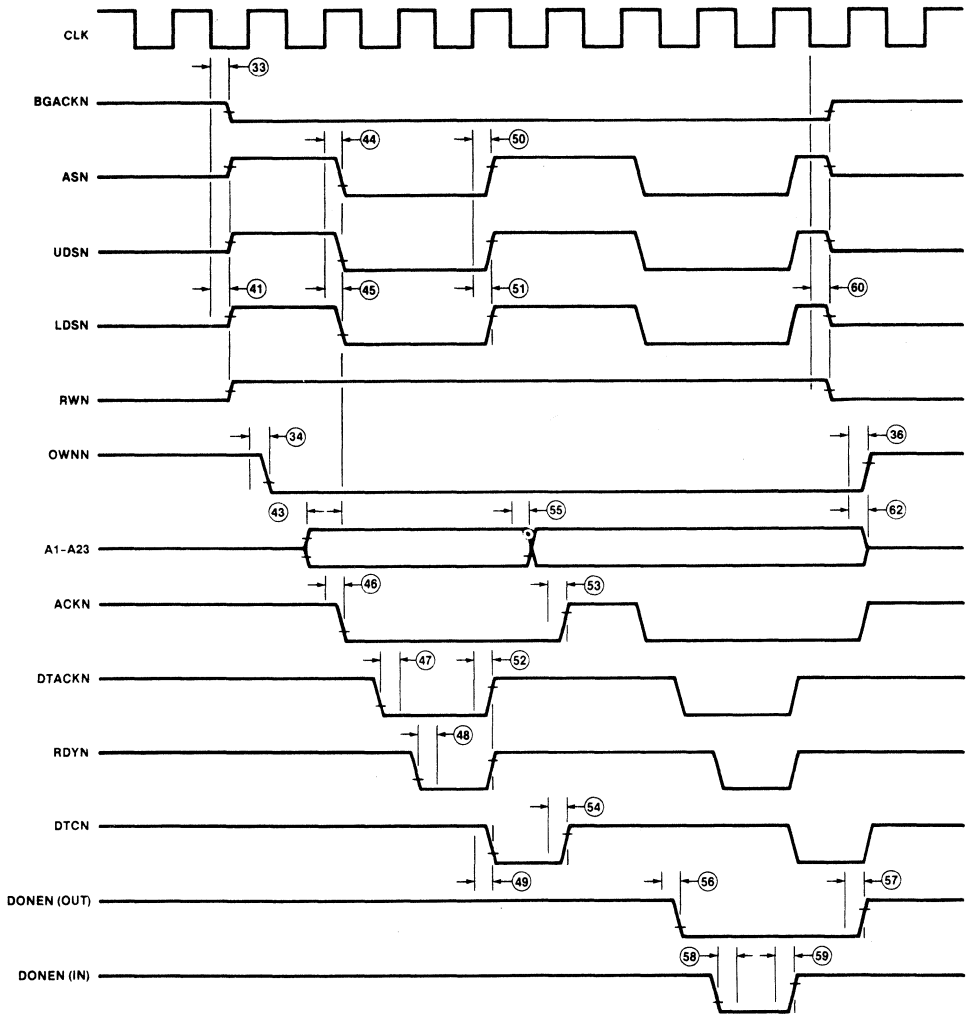


Figure 11. Read from Memory, Write to Device

NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Preliminary

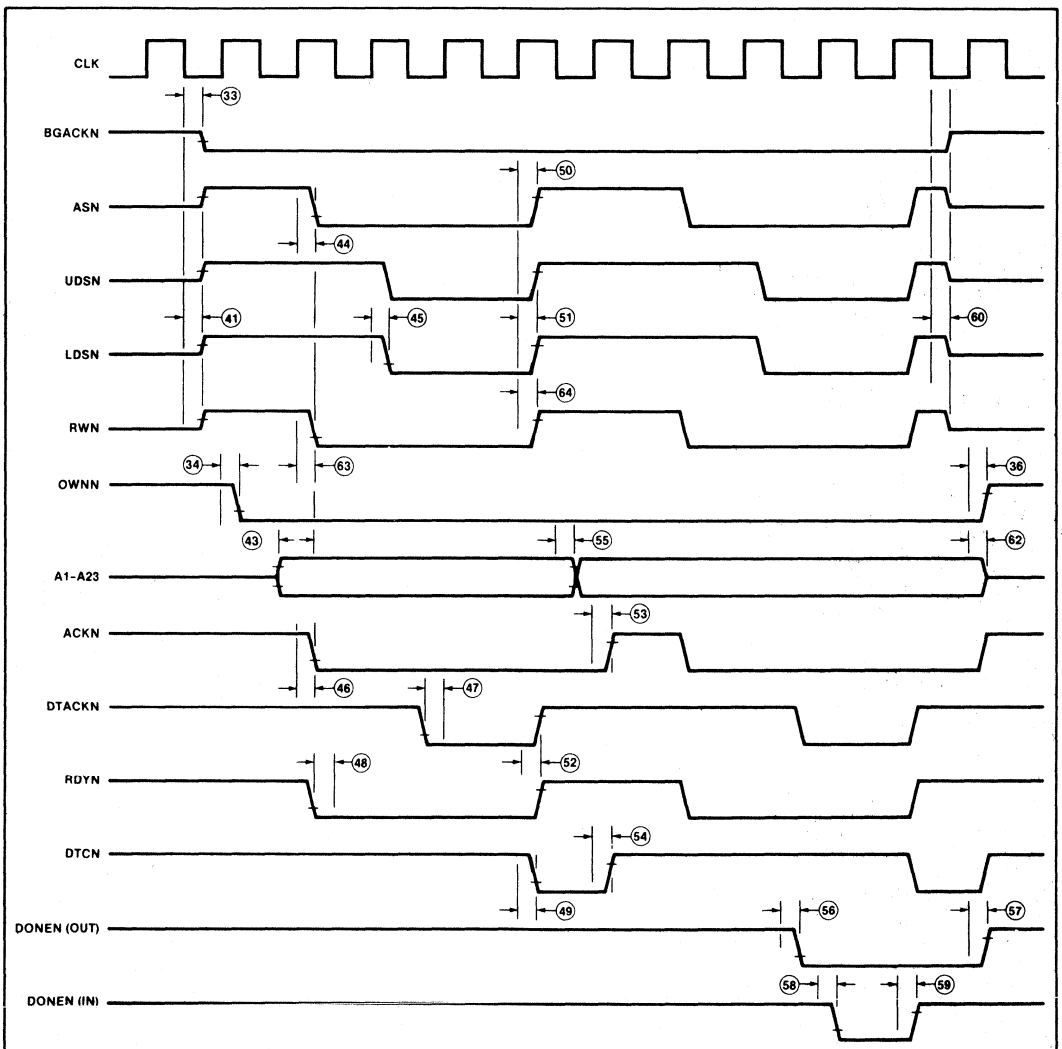


Figure 12. Write to Memory, Read from Device

NOTE:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.

Preliminary

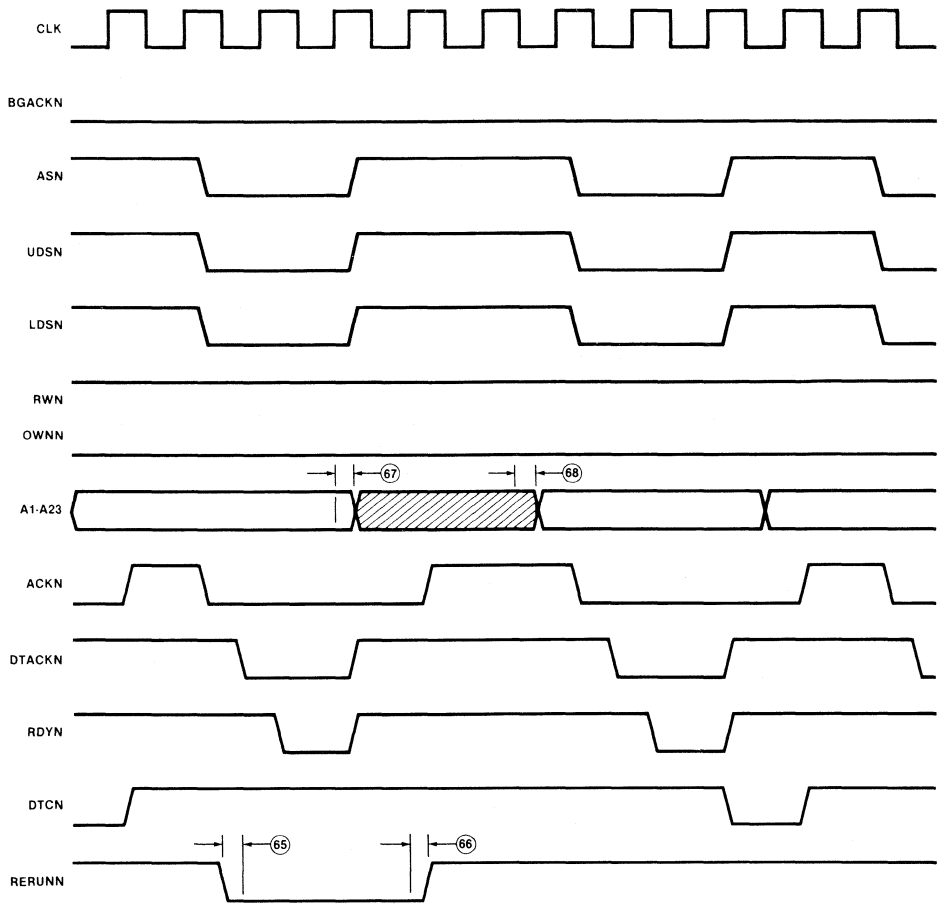


Figure 13. Rerun Asserted During Read from Memory, Write to Device

NOTES:
1. 16-bit transfer illustrated. For 8-bit transfer either LDSN or UDSN, but not both, will be asserted each cycle, depending on byte address.
2. DMAI will release the bus after a RERUNN if there is no valid request. The next request will then retry the cycle which was terminated by the RERUNN signal.
3. RERUNN must be asserted no later than DTACKN and RDYN.
4. If a cycle is terminated by RERUNN, the transfer count will be one less than the actual data transferred correctly. The double RERUNN signal on the same cycle will terminate the DMAI operation with a status bit set and an interrupt generated (if enabled).

Made to interface directly with systems based on a 68000 processor, a single-chip DMA controller unclogs memory-to-peripheral data transfers.

DMA controller meshes with 68000 systems

Moving large blocks of data between peripherals and system memory without the intervention of the system CPU, direct memory access is a proven scheme for bettering I/O operations and raising system throughput. To complement the high speed of the 68000 microprocessor, a single-channel DMA controller, or interface chip, is both fast and compatible with the 68000's architecture and bus structure. Such compatibility considerably reduces the required system hardware design efforts.

The basic function of the chip, the SCB68430, is to transfer a series of operands (data) between system memory and a peripheral device. Operands can be in the form of bytes, words (16 bits) or long words (32 bits). Users can program the chip to transfer data in either a single cycle (called cycle stealing) or a burst mode. In the burst mode, the device can transfer blocks of up to 64 kbytes.

Because of the operating speed of 68000-based systems, the DMA controller is fabricated using integrated Schottky logic. Current versions operate at 8 or 10 MHz, but I^2L technology makes possible faster parts, which will be out later this year. At any of those speeds, the controller helps to eliminate the memory-to-I/O bottlenecks prevalent in conventional memory-handling systems.

Table 1 lists typical data transfer rates for the DMA controller as a function of clock frequency. Notice the last two columns, which show read and write times achievable over a 32-bit VMEbus. At any clock frequency, use of the VMEbus results in transfer rates at least twice as fast as those over the 68000 bus.

Guy Thomsen, S68000 Supervising Engineer
Signetics Corp.
PO Box 409, Sunnyvale, Calif. 94086.

The controller's internal structure is divided into three sections, called the device interface, the host interface, and the internal register set (Fig. 1). The device interface consists of control logic that initiates a DMA transfer between a peripheral device and memory. Once the chip's internal register set has been initialized, the Request signal (REQ) starts the DMA process. Along with REQ, the Acknowledge (ACK), Ready (RDY), and Device Transfer Control (DTC) lines all provide handshake signals that allow a wide variety of peripherals to interface with the DMA circuit. The bidirectional line Done, normally used by the controller to inform a peripheral that a data transfer is complete, can also be used by the peripheral to inform the controller that transfer operations will be terminated.

Since the host interface contains all 68000 control signals, the chip can arbitrate for the system bus and transfer data over the bus. Also in the host interface is the address decoder for its register set. The chip's logic matches the 68000's vectored interrupt structure. That is, after an interrupt request, the interrupt acknowledgment circuit puts a programmable interrupt vector on the bus for the system to use.

Within the controller's register structure are the control and status registers, two programmable counters, and the interrupt logic register. There also are inactive registers that serve to maintain compatibility with other 68000 DMA controllers (the 68440 and the 68450). All registers are addressable with 8-bit words. To simplify operation with the 68000's MOVEP instruction, addresses are ordered so that certain sets of registers can be accessed as words or as long words (see Table 2).

For compatibility with the 68440 and 68450 DMA controllers, the 68430's control and status bits are

Reprinted from **ELECTRONIC DESIGN- October 27, 1983**

mapped in positions equivalent to those in the register maps of those two devices. Bits used in the other controllers but not in the 68430 are assigned default values. If a designer requires upward compatibility with these controllers, he uses the default values when writing control words to the registers, although they have no effect on the 68430. When a register is read, the default value is returned regardless of the originally programmed value.

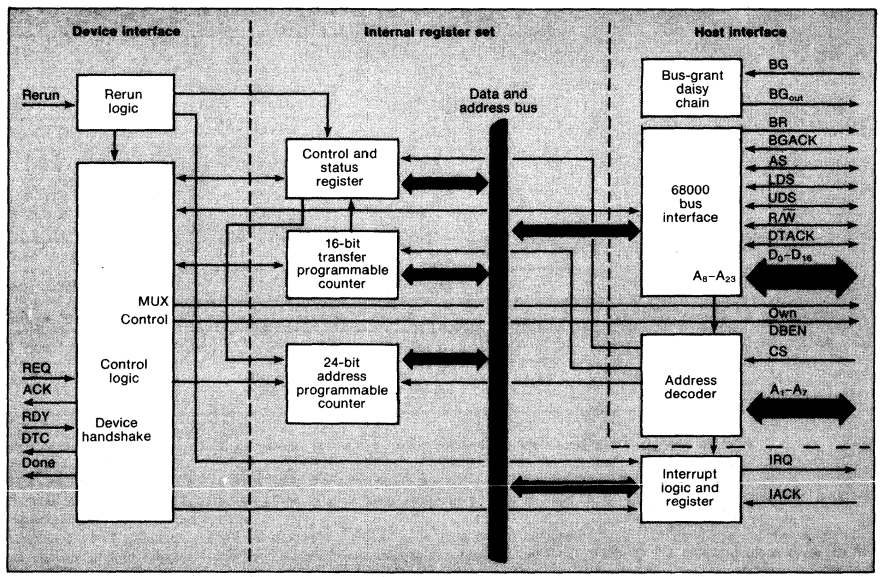
A look at the registers

An External Request Mode bit in the device control register determines whether the chip operates in the burst or the cycle-stealing mode. The burst mode allows a peripheral to request the transfer of multiple operands on consecutive bus cycles. In the cycle-stealing mode, a peripheral requests an operand transfer by generating a falling-edge signal on REQ. The controller services the request by arbitrating for the bus, obtaining the bus, and notifying the peripheral by asserting the ACK output.

The chip's operation control register selects the direction of operand transfers and the size of the operands. A transfer can be from system memory to a peripheral or from a peripheral to memory. The operand size bits determine whether an Upper Data Strobe (UDS), a Lower Data Strobe (LDS), or both are to be generated during the transfer cycle and the increment by which the memory address counter is changed in each transfer cycle. Operand sizes can be specified as a byte, a 16- bit word, a long word (separate transfers of two words) or a double word (a 32-bit transfer).

The memory address counter is incremented by 1 after each byte-operand transfer and by 2 after each word-operand transfer. When the operand size is a long word, the operand is transferred as two 16-bit words. In that case, the counter is incremented by 2 after the transfer of each 16-bit word. A double-word operand is transferred as a single 32-bit word, and the counter is incremented by 4 after each operand transfer.

The sequence control register serves no internal function but is needed for compatibility with the programming of the 68440 and 68450 controllers. The same is true of the channel priority register.



1. To connect to a 68000-based system, the SCB68430 DMA controller needs two interface blocks—device and host—and an internal register set. All 68000 control signals are included in the host interface, and the device interface controls all DMA transfers between a peripheral and memory.

Contained in the channel control register are the Start Operation bit, the Software Abort bit and the Interrupt Enable bit. Setting the Start bit initiates operation of the controller, the Software Abort bit terminates the controller's current operation and places it in the idle state, and the Enable Interrupt bit generates an Interrupt Request if the Channel Operation Complete bit in the channel status register is set. The Channel Operation Complete bit is set following the termination of any of the chip's operation and indicates that a DMA transfer is complete—it is set regardless of whether the operation was successful—and must be cleared before another channel operation can start.

Reading the channel status register reveals the controller's status. The Channel Operation Complete (COC), Normal Device Terminate (NDT), and Error (ERR) bits can be cleared by writing a logical 1 to the appropriate register bit positions. Bit positions written with a logical 0 remain unaffected.

After a channel is started, the Channel Active bit is set and remains so until channel operation ends. The Normal Peripheral Device Termination bit is set when the peripheral ends DMA operation. The peripheral asserts the Done signal while it is being acknowledged. The termination bit must be cleared before another channel operation can begin. When the channel status register is read, the Ready Input

Table 1. Transfer rates for the SCB68430

Clock frequency (MHz)	Mode	From memory to device (Mbits/s)	Mode	To memory from device (Mbits/s)	Read 32-bit word over VMEbus (Mbits/s)	Write 32-bit word over VMEbus (Mbits/s)
8	Byte	2	Byte	1.6	8	6.4
	Word	4	Word	3.2		
	Long word	4	Long word	3.2		
10	Byte	2.5	Byte	2	10	8
	Word	5	Word	4		
	Long word	5	Long word	4		
12.5	Byte	3.125	Byte	2.5	12.5	10
	Word	6.25	Word	5		
	Long word	6.25	Long word	5		

Table 2. Address map for the SCB8430

Address bits								Register		Mode	Affected by Reset
7	6	5	4	3	2	1	0	Name	Acronym		
d	d	0	0	0	0	0	0	Channel status register	CSR	R/W	Yes
d	d	0	0	0	0	0	1	Channel error register	CER	R	Yes
d	d	0	0	0	0	1	0	Reserved			
d	d	0	0	0	0	1	1	Reserved			
d	d	0	0	0	1	0	0	Device control register	DCR	R/W	Yes
d	d	0	0	0	1	0	1	Operation control register	OCR	R/W	Yes
d	d	0	0	0	1	1	0	Sequence control register	SCR	R/W	Yes
d	d	0	0	0	1	1	1	Channel control register	CCR	R/W	Yes
d	d	0	0	1	0	0	0	Reserved			
d	d	0	0	1	0	0	1	Reserved			
d	d	0	0	1	0	1	0	Memory transfer counter high	MTCH	R/W	No
d	d	0	0	1	0	1	1	Memory transfer counter low	MTCL	R/W	No
d	d	0	0	1	1	0	0	Memory address counter high	MACH	R/W	No
d	d	0	0	1	1	0	1	Memory address counter middle high	MACMH	R/W	No
d	d	0	0	1	1	1	0	Memory address counter middle low	MACML	R/W	No
d	d	0	0	1	1	1	1	Memory address counter low	MACL	R/W	No
d	d	0	1	d	d	d	d	Reserved			
d	d	1	0	0	0	d	d	Reserved			
d	d	1	0	0	1	0	0	Reserved			
d	d	1	0	0	1	0	1	Interrupt vector register	IVR	R/W	Yes
d	d	1	0	0	1	1	0	Reserved			
d	d	1	0	0	1	1	1	Interrupt vector register	IVR	R/W	Yes
d	d	1	0	1	0	d	d	Reserved			
d	d	1	0	1	1	0	0	Reserved			
d	d	1	0	1	1	0	1	Channel priority register	CPR	R/W	No
d	d	1	0	1	1	1	0	Reserved			
d	d	1	0	1	1	1	1	Reserved			
d	d	1	1	d	d	d	d	Reserved			

d = don't care

State bit reflects the state of the RDY input. This bit is a logical 0 if RDY is low and a logical 1 if it is high. Write or reset operations have no effect on the bit.

If the controller's operation is terminated because of an error, the setting of the Error bit indicates this fact. The channel error register contains a field that indicates the source of an error. The register's contents are cleared when bit 12 is cleared. Examples of field indications are 00000 for no error, 01001 for a bus error that occurred during the last bus cycle generated by the controller, and 10001 for termination of channel operation by an Abort command.

A 32-bit memory address counter (MACH-MACMH-MACML-MACL) specifies either the memory location holding the first operand that is to be transferred or the memory location the first operand is to be transferred to, depending on the direction of the transfer. This counter must be initialized before transferring of a data block, and then it increments automatically, depending on the operand length.

Only the least significant 24 bits of the counter are implemented in the controller—these are MACMH, MACML and MACL. The most significant byte, MACH, is included just to provide compatibility with the 68440 and 68450. Writing to MACH has no effect on the chip's operations, whereas reading MACH always returns the hexadecimal value 00.

A 16-bit memory transfer counter programs the number of operands to be transferred. The counter must be initialized before starting a data-block transfer, and then it decrements once for each operand transfer—regardless of operand size—until it reaches zero. Channel operation then terminates and the COC bit in the channel status register is asserted.

An interrupt vector register contains the value to be placed on the data bus upon receipt of an interrupt acknowledgment from the controlling microprocessor. The controller uses only the seven most significant bits of the programmed value. Its output vector contains a logical 0 in the LSB position if a normal termination occurred (the Error bit was not set) and a logical 1 if termination was caused by an error (the Error bit was set).

The contents of the interrupt vector register are initialized to OF_{16} by a Reset signal. The value returned is OF_{16} regardless of the error state, and this value remains until the register is programmed by the controlling microprocessor. For compatibility with the 68440 and 68450, the interrupt vector register has two addresses—one normal and one for an error (see Table 2 again). If program compatibility with the other DMA controllers is required, the value written to the normal address must have a logical 0 as its LSB, and the value at the error address is the same except that the LSB must be a logical 1.

Step by step

Upon receiving a valid request for a data transfer from a peripheral device, the controller arbitrates for, and obtains ownership of, the system bus. It indicates that it desires to be the bus master by asserting its Bus Request line (BR), a wired-OR signal that indicates to the processor that an external device requires control of the bus.

The processor is at a lower priority level than external devices and will give up the bus after completing the last bus cycle it has started. It puts the bus up for external arbitration by asserting its own Bus Grant output (BG). This signal can be routed through a daisy chain—such as provided by the controller—or through another priority-encoded network. When the controller receives the Bus Grant—its BG input is enabled—it becomes the next bus master. The controller waits until the Address Strobe (AS), Data Transfer Acknowledge (DTACK), and Bus Grant Acknowledge (BGACK) signals become inactive before assuming bus ownership by asserting its own BGACK line. Next, the controller de-activates its BG line and proceeds to transfer data. At the completion of this phase, it gives up bus ownership by de-activating its BGACK output.

In the burst mode, the detection of an active-low request after the controller's operations have begun results in the start of the bus arbitration cycle. But if a peripheral de-activates its REQ line at least one clock cycle before the controller asserts BGACK, the chip de-activates its Bus Request line and will not assume ownership of the bus.

The Interrupt Enable bit—bit 3 in the channel control register—determines whether the controller generates interrupt requests. When the bit is set, an Interrupt Request (IRQ) is generated if the Channel Operation Complete bit in the channel status register is set. When the Interrupt Acknowledge input (ITACK) is asserted and the DMA controller has an interrupt request pending, it returns an interrupt vector on the data bus.

Shaking hands

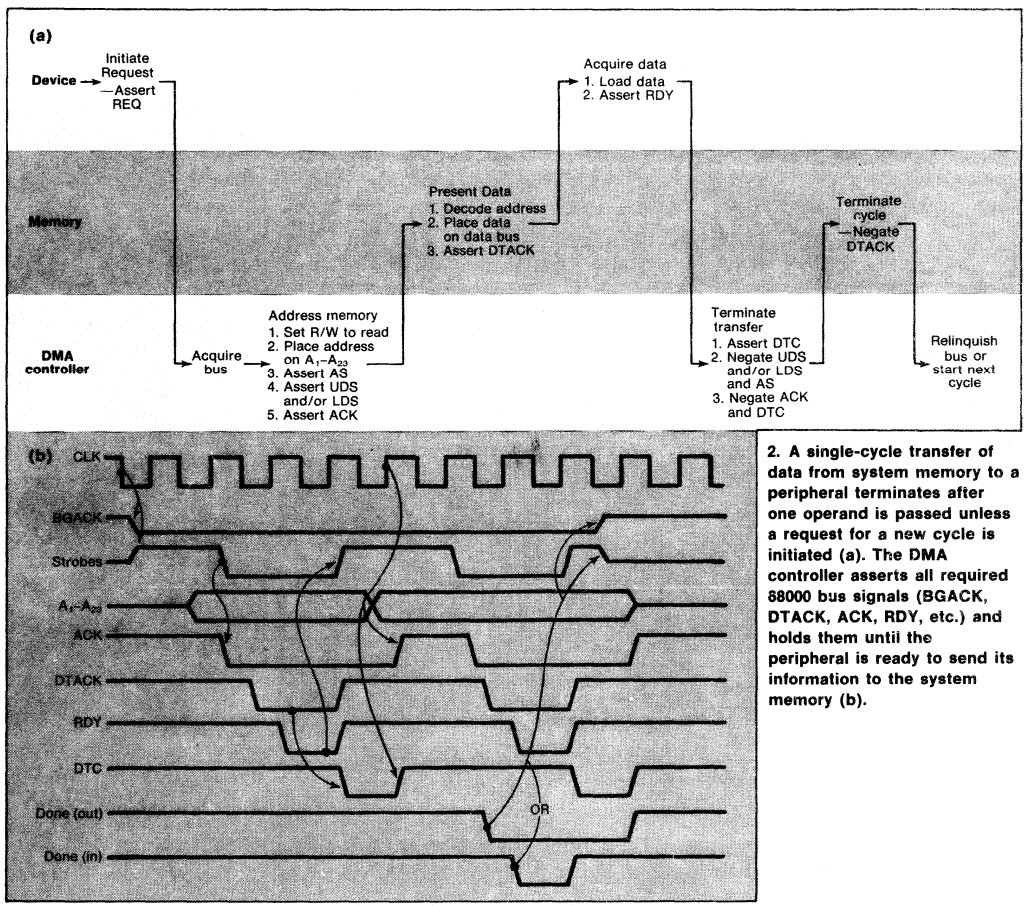
A peripheral makes a request for service by asserting its Request line. The controller can then operate in either the burst or the cycle-stealing mode, as programmed by the External Request Mode bit. When the controller asserts its Acknowledge line, it implicitly addresses the peripheral making the request during data transfers. RDY is an active-low

input from the requesting peripheral that tells the controller that valid data has been either stored or put on the bus.

Done is a bidirectional active-low signal. As an output, it is asserted and negated by the controller concurrently with the acknowledgment output of the last operand. This indicates to the peripheral that the memory transfer count is exhausted and that the controller's operation is complete as a result of that transfer. DTC is an active-low output asserted by the chip to signal the peripheral that the requested data transfer is complete. On a write-to-memory operation, it indicates that the data provided by the peripheral has been sucessfully stored. On a read-from-memory operation, DTC indicates to the peripheral that data from memory is present on the bus and should be latched. DTC is not asserted if assertion of the rerun input terminates the bus cycle.

An actual transfer of data between memory and a peripheral occurs during the data transfer phase (Fig. 2a). All transfers occur during a single cycle except for long-word operands, which require two cycles—the operand is transferred as two 16-bit words. Transfers take place using a so-called single-address protocol; the controller addresses memory via the bus address lines, and the peripheral is addressed implicitly via ACK.

When a Request is generated using the request method programmed in the control register, the controller obtains the bus and asserts ACK to notify the peripheral that a transfer is about to take place. It also asserts all 68000 bus control signals needed for the transfer (Fig. 2b) and holds them until the peripheral responds with RDY. The bus cycle then terminates normally. RDY can be tied low (activated) if the peripheral is fast enough. On transfers from memory to a peripheral, data is valid when



2. A single-cycle transfer of data from system memory to a peripheral terminates after one operand is passed unless a request for a new cycle is initiated (a). The DMA controller asserts all required 68000 bus signals (BGACK, DTACK, ACK, RDY, etc.) and holds them until the peripheral is ready to send its information to the system memory (b).

DTACK is asserted by the memory and remains valid until the data strobes are de-activated.

When a transfer is from a peripheral to memory (Fig. 3a), data must be valid on the bus before the controller asserts the data strobe or strobes. The peripheral indicates valid data by asserting its RDY line. The controller then asserts the strobes and holds them in this state until the memory accepts the data, indicated by the assertion of DTACK (Fig. 3b). The controller then asserts DTC to de-activate the data strobes.

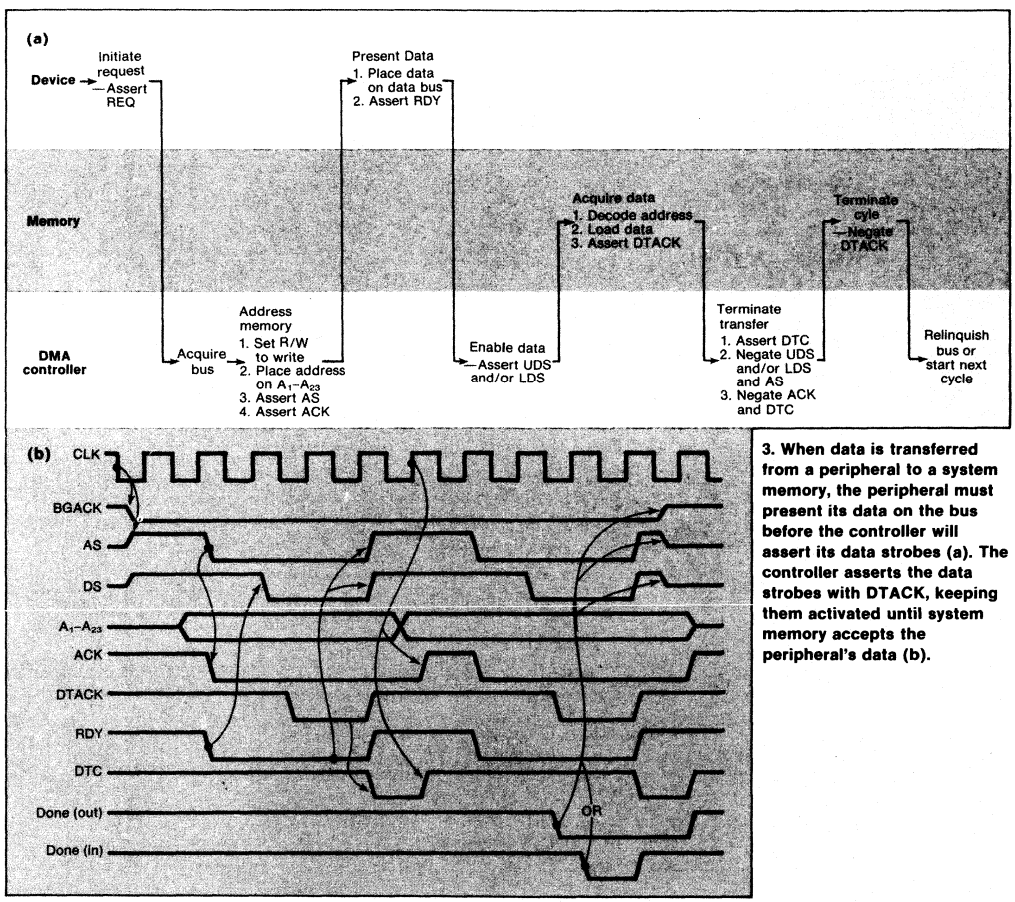
DMA capability for a dual UART

In a communications application, the controller can furnish DMA capability for an SCN68681 dual universal asynchronous receiver-transmitter, a single chip that provides two independent full-duplex asynchronous receiving or transmitting channels.

The controller contains all signals necessary to interface the dual UART as both a peripheral and a system master (Fig. 4). Although its control signals drive the 68000 bus directly, devices U_1 through U_5 are required for demultiplexing its address/data bus lines. U_1 is an input buffer for the register address. It and octal buffer and line drivers U_2 and U_3 are active during the DMA operation and drive the memory address signals onto the bus. During a register read/write cycle, octal bus transceivers U_4 and U_5 serve as bidirectional data buffers.

All the other circuitry shown is used for interfacing the dual UART. Address lines to the UART come from multiplexer U_6. The output is selected by the Own signal from the controller, asserted during DMA transfers. The selection is among the address bus, UDS, and a hard-wired address.

The UART is normally addressed at either even



3. When data is transferred from a peripheral to a system memory, the peripheral must present its data on the bus before the controller will assert its data strobes (a). The controller asserts the data strobes with DTACK, keeping them activated until system memory accepts the peripheral's data (b).

or odd memory locations, but it can be addressed at contiguous address spaces. The latter is accomplished by ANDing the UART selection line with either LDS or UDS (U_8 and U_9) to generate the receiver-transmitter's CS signal and using UDS for its least significant address.

The hard-wired address, selected during DMA operations, specifies the receiver-transmitter registers. The MSB of this address, from flip-flop U_7, determines the channel to be accessed. The flip-flop is set by the system during initialization.

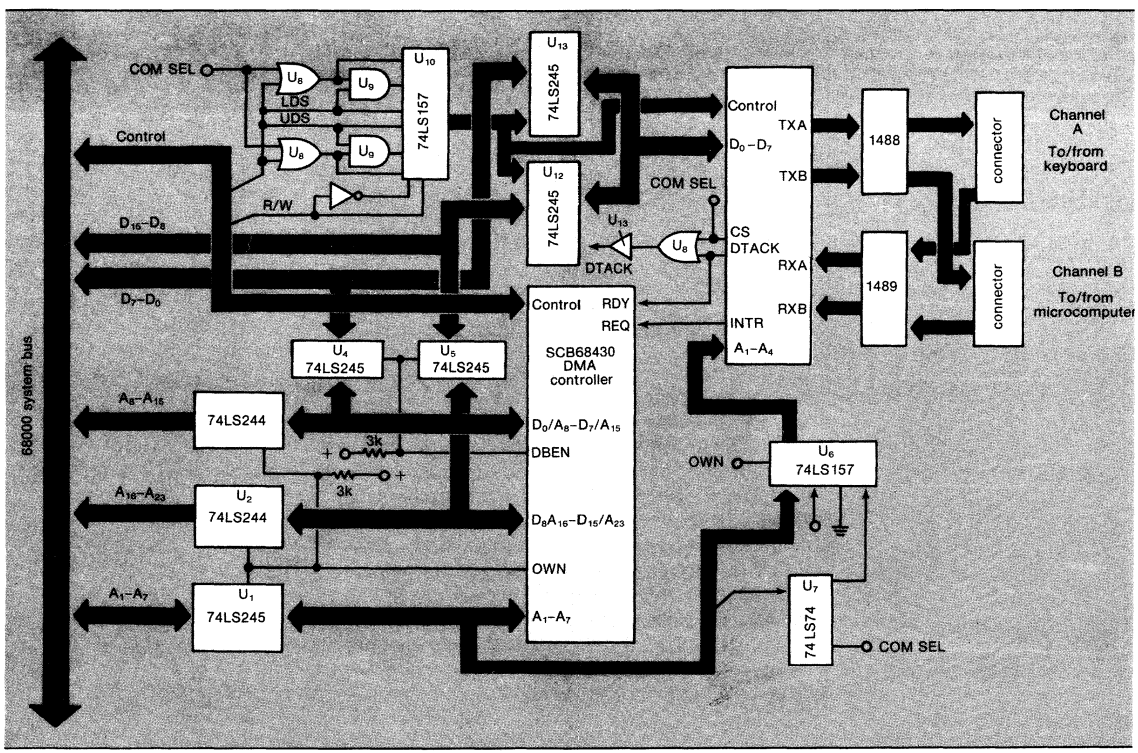
Making a path to the UART

To control the data path to the UART during system and DMA accesses, multiplexer U_{10} is used to gate the UART's CS and R/W signals. ANDed signals from gates U_8 and U_9 also enable the proper transceiver buffer for data to or from the UART through U_{10}. While the controller is operating, R/W is inverted to the UART, since a read from memory is a write to the receiver-transmitter. During host access, the system requires that a DTACK signal be generated at the proper time. This signal is gated onto the bus by U_8 and U_{13}.

The UART's Interrupt Request line (INTR) is used to request DMA operation by being routed to the controller's REQ input. When a DMA transfer is in progress, the DTACK signal from the UART is used as the RDY signal for the controller.

Before the UART or the controller can transfer data, both must be initialized. For the UART, that means setting the communication parameters (see Table 3). In this application, the controller's interrupt vector register is also loaded during initialization. Channel A is primarily an output port to a display monitor, and channel B is an input port from a small microcomputer. In both cases, the controller transfers information in block form.

After the system software determines that information is to be transferred, it sets a bit in memory and calls the driver (see Table 4). The driver examines the bit to determine which channel is to be



4. To make a dual UART DMA-compatible, the DMA controller interfaces the device with the 68000 bus and allows it to receive and transmit data to the outside world.

Table 3. Initialization routine

```
COMINIT  MOVE.B #$10, CRA ; RESET POINTER
         MOVE.B #$02, MR1A ; 7 BITS, EVEN PARITY
         MOVE.B #$37, MR2A ; ENABLE RTS, CTS
         MOVE.B  #$44, CSRA  ; SET 300 BAUD RATE
         MOVE.B  #$30, CRA   ; RESET.TRANSMITTER-RECEIVER
         MOVE.B  #$20, CRA   ; NOW ENABLE BOTH
         MOVE.B  #$05, CRA   ; NOW ENABLE BOTH

         MOVE.B  #$10, CRB   ; RESET POINTER
         MOVE.B  #$02, MR1B  ; 7 BITS, EVEN PARITY
         MOVE.B  #$37, MR2B  ; ENABLE RTS, CTS
         MOVE.B  #$44, CSRB  ; SET 300 BAUD RATE
         MOVE.B  #$30, CRB   ; RESET TRANSMITTER-RECEIVER
         MOVE.B  #$20, CRB   ;
         MOVE.B  #$05, CRB   ; NOW ENABLE BOTH

         MOVE.B  #$33, ISR   ; SET INTERRUPT CONDITIONS

         MOVE.B  #$72, IVR   ; SET INTERRUPT TO VECTOR POINT

         RTS                 ; WE ARE DONE
```

Note: Channel A is the monitor and channel B the microprocessor.

Table 4. A simple driver

```
TRANSFER MOVE.B  DO,CSRA     ; RESET A3 AT FLIP-FLOP U7
         BTST.B  #00,CHAN    ; CHANNEL _ _ IS TO BE ACTIVE
         BEQ     ONWARD      ; SKIP SETTING OF BIT IF
                               CHANNEL A

         MOVE.B  DO,CSRB     ; SET A3 AT THE FLIP-FLOP U7

ONWARD   MOVE.W  LENGTH,     ; SET THE TRANSFER LENGTH
                 MTCH
         MOVE.L  BUFFER,     ; LOAD THE BUFFER ADDRESS
                 MACH
         MOVE.L  TPXFER,     ; SET THE TRANSFER MODE,
                 DCR           DIRECTION, AND START BIT

         OPERATION NOW STARTED, SO RETURN TO MAIN
         PROGRAM

         RTS                 ; GOOD-BYE
```

activated. It then either sets or resets the MSB of the UART's address (flip-flop U_7). Next, the driver loads the controller. The manner in which the controller's registers are arranged enables the software driver to be very simple. For example, the transfer driver need only contain three instructions to set the operand transfer length and the system buffer address, and to load the transfer type and start bit. The three variables that are required for that sequence can be locations in system memory that can be updated or changed as necessary by the supervisory program. □

Disk control

Intelligent Multiple Disk Controller (IMDC)

Originally published by Signetics February 1985

Preliminary

DESCRIPTION

The SCN68454 Intelligent Multiple Disk Controller (IMDC) provides the traditional and advanced features required to control Winchester type rigid disks and floppy disks. It can control up to four rigid or floppy disk drives, in any combination. It can be used to control, with a minimum of external hardware, any drive that has an SA1000 interface standard or an ST506 Seagate interface standard.

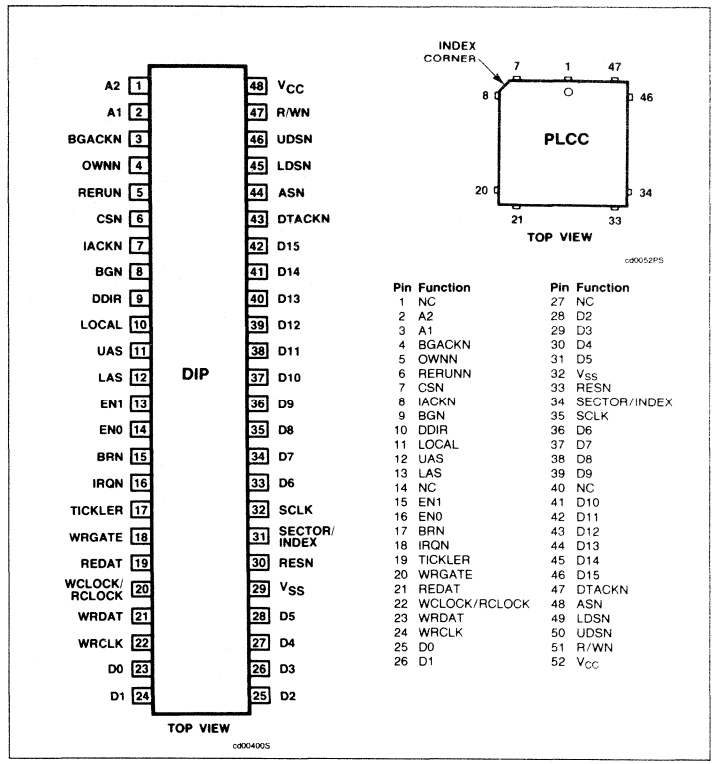
The IMDC supports soft or hard sectored disk track format and standard IBM track formats for floppy disks (both single and double density).

The controller is programmable via an external host processor by the use of high level commands. Data transfer on the host data bus can be 8 or 16-bit in parallel. The IMDC is capable of handling a serial data rate of up to 10Mbits per second. The SCN68454 is constructed using Signetics MOS-VLSI technology.

FEATURES

- **Bus compatible with SCN68000 microprocessor**
- **Automatic rerun on bus error**
- **Supports SCN68000 vectored interrupts**
- **31-bit address counter**
- **16/8-bit data transfers**
- **Supports up to 4 rigid disks and floppy disks in any combination**
- **Supports SA1000 and ST506 Winchester interfaces**
- **Data rates up to 10Mbits per second for MFM**
- **Data rates up to 2Mbits per second for FM**
- **Handles FM and MFM data encoding/decoding**
- **On chip DMA controller and FIFO buffer (128 bytes)**
- **Multiple sector read/write with implied seek**
- **Automatic bad sector handling**
- **32 and 40-bit ECC programmable polynomials**
- **Supports computer generated ECC polynomials**

PIN CONFIGURATION



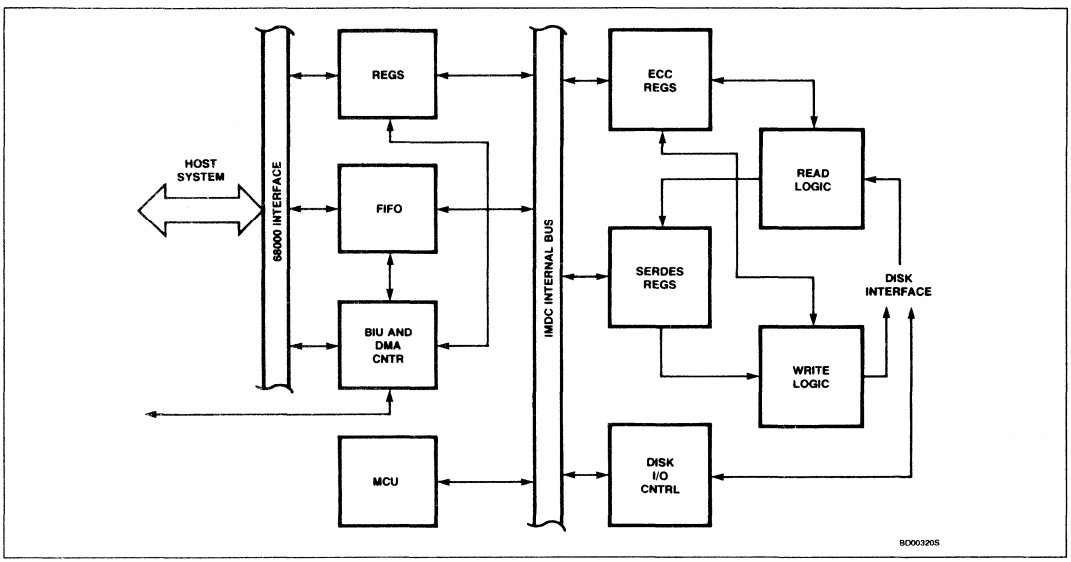
Pin	Function	Pin	Function
1	NC	27	NC
2	A2	28	D2
3	A1	29	D3
4	BGACKN	30	D4
5	OWNN	31	D5
6	RERUNN	32	V_{SS}
7	CSN	33	RESN
8	IACKN	34	SECTOR/INDEX
9	BGN	35	SCLK
10	DDIR	36	D6
11	LOCAL	37	D7
12	UAS	38	D8
13	LAS	39	D9
14	NC	40	NC
15	EN1	41	D10
16	EN0	42	D11
17	BRN	43	D12
18	IRQN	44	D13
19	TICKLER	45	D14
20	WRGATE	46	D15
21	REDAT	47	DTACKN
22	WCLOCK/RCLOCK	48	ASN
23	WRDAT	49	LDSN
24	WRCLK	50	UDSN
25	D0	51	R/WN
26	D1	52	V_{CC}

Preliminary

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0 to 70°C
Ceramic DIP	SCN68454CGI48
Plastic DIP	SCN68454CGN48
Plastic LCC	SCN68454CGA52

BLOCK DIAGRAM



PIN DESCRIPTION

The pin description table describes the function of each of the pins of the IMDC. Signal names ending in 'N' are active low. All other signals are active high. In the descriptions, 'REG mode' refers to the state when the IMDC is chip selected. The term 'DMA mode' refers to the state when the IMDC assumes ownership of the bus. The term 'LOCAL mode' refers to the state when the IMDC is transferring data to or from the disk interface. The IMDC is in the 'IDLE mode' at all other times.

In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic zero) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
A1, A2	1, 2	2, 3	I/O	**Address Lines:** Active high, three-statable. In the REG mode, these low order address lines specify which internal register of the IMDC is being accessed. In DMA mode, A1 – A2 are outputs which provide the low order address bits of the location being accessed. Three-stated in IDLE and LOCAL mode.
D0 – D15	23 – 28, 33 – 42	25, 26, 28 – 31, 36 – 39, 41 – 46	I/O	**Data Lines:** Active high, three-statable. In REG mode, the bidirectional data lines are used to transfer data between the CPU and the IMDC registers. In LOCAL mode, the bidirectional data lines are used to transfer data between the IMDC and the disk unit (three-stated in IDLE mode).
ASN	44	48	I/O	**Address Strobe:** Active low, three-statable. In REG and IDLE modes, ASN is an input which indicates that the current bus master has placed a valid address on the bus. It is monitored by the IMDC during bus arbitration to ascertain that the previous bus master has completed the current bus cycle. In DMA mode, it is an output indicating that the IMDC has placed a valid address on the bus.

Preliminary

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
UDSN	46	50	I/O	**Upper Data Strobe:** Active low, three-statable. In REG and IDLE modes, UDSN is an input which indicates that the upper data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning. In an eight bit system this input is tied to DSN.
LDSN	45	49	I/O	**Lower Data Strobe:** Active low, three-statable. In REG and IDLE modes, LDSN is an input which indicates that the lower data byte of the addressed word is being addressed. In DMA mode, it is an output with the same meaning. In an eight bit system this input is tied to A0.
R/WN	47	51	I/O	**Read/Write:** Active high for read, low for write, three-statable. In REG mode, R/WN is an input which controls the direction of data flow through the IMDC's input/output data bus interface and through an external data bus buffer. R/WN high causes the IMDC to place the data from the addressed register on the data bus, while R/WN low causes the IMDC to accept data from the data bus. In DMA mode, R/WN is an output to memory and I/O controllers indicating the type of bus cycle. It is held three-stated during IDLE and LOCAL mode.
CSN	6	7	I	**Chip Select:** Active low. When low, places the IMDC into the REG mode. This input signal is used to select the IMDC for register data transfers. These transfers take place over the D0 – D15 lines as controlled by the R/WN and A1 – A2 inputs. The IMDC is deselected when CSN is high. CSN is ignored during DMA mode.
DTACKN	43	47	I/O	**Data Transfer Acknowledge:** Active low, three-statable. In REG mode, DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate that valid data is present on the bus. The signal is negated (driven high) when completion of the cycle is indicated by negation of the CSN or IACKN input. In DMA mode, DTACKN is an input monitored by the IMDC to determine when the addressed device (memory) has latched the data (write cycle) or put valid data on the bus (read cycle).
RERUNN	5	6	I	**Rerun:** Active low. This input is asserted by external error detect logic to indicate a bus error. In DMA mode, the IMDC stops operation and three-states the data, address, and control lines, except BGACKN. It remains IDLE until RERUNN becomes inactive, and then retries the last bus cycle. If RERUNN is asserted again, the IMDC sets the error code in the main status byte, stops DMA operation, releases the bus, and interrupts the CPU. Not monitored in REG, LOCAL and IDLE modes.
RESN	30	33	I	**Master Reset:** Active low. Assertion of this pin clears the internal registers and initializes the interrupt vector register to H'0F'. All bidirectional I/O lines are three-stated and the IMDC is placed in the IDLE mode.
SCLK	32	35	I	**Clock:** Active high. Usually the system clock, but may be any clock meeting the electrical specifications. Used by the IMDC to synchronize disk functions and external control lines, and may not be gated off at any time. The frequency should be 16MHz ± 1%.
IRQN	16	18	O	**Interrupt Request:** Active low, open drain. This output is asserted at the end of each command execution. The CPU can read the status register to determine the interrupting condition, or can respond with an interrupt acknowledge cycle to cause the IMDC to output an interrupt vector on the data bus.
IACKN	7	8	I	**Interrupt Acknowledge:** Active low. When asserted, indicates that the current cycle is an interrupt acknowledge cycle. The IMDC normally responds by placing the contents of the interrupt vector register on the data bus and asserting DTACKN.
BRN	15	17	O	**Bus Request:** Active low, open drain. BRN is asserted by the IMDC to request ownership of the bus for a DMA transfer. It is negated when the bus has been granted (BGN low) and BGACKN has been asserted.
BGN	8	9	I	**Bus Grant:** Active low. BGN indicates to the IMDC that it is to be the next bus master. After BGN is asserted, the IMDC waits until DTACKN, ASN, and BGACKN have become inactive before assuming ownership of the bus by asserting BGACKN.
BGACKN	3	4	I/O	**Bus Grant Acknowledge:** Active low, open drain. As an input, BGACKN is monitored by the IMDC during the bus arbitration cycle to determine when it can assume ownership of the bus (BGACKN negated). In DMA mode, it is asserted by the IMDC to indicate that it is the bus master. Three-stated in REG, LOCAL, and IDLE modes.
EN0	14	16	O	**Enable 0:** Active high. This signal is asserted during LOCAL mode when the IMDC transmits disk control signals on the data lines D0 – D15. The assertion or negation of EN0 is used to latch the control signals at the disk interface.
EN1	13	15	O	**Enable 1:** Active low. This signal is asserted during LOCAL mode when the IMDC reads the status of the disk drive. The assertion of EN1 is used to enable the status information onto data lines D0 – D15.

Preliminary

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
UAS	11	12	O	**Upper Address Strobe:** Active high. UAS is active only during DMA mode. It is used to latch the upper address bits A19 – A31 from the address/data lines.
LAS	12	13	O	**Lower Address Strobe:** Active high. LAS is active only during DMA mode. It is used to latch the lower address bits A3 – A18 from the address/data lines.
REDAT	19	21	I	**Composite Read Data:** This signal is the composite disk data synchronized to the RCLOCK signal generated by the external PLL.
WRGATE	18	20	O	**Write Gate:** Active high. This signal is asserted during disk write operations. Disasserted during all other modes.
WCLOCK/ RCLOCK	20	22	I	**Write Clock/Read Clock:** This clock input is generated by external logic such as the SCB68459 DPLL. During disk write operations, the input is defined as WCLOCK which provides the bit-cell frequency to the IMDC. The input is defined as RCLOCK during disk read operations. RCLOCK is twice the data frequency and is synchronized to REDAT.
WRDAT	21	23	O	**Write Data Pattern:** Active high. This signal provides the write data pattern for external logic, like the SCB68459 DPLL, to use with WRCLK and WCLOCK to generate the write data pulse stream to a disk unit. WRDAT is an non-return to zero (NRZ) signal which changes state on the rising edge of WCLOCK.
WRCLK	22	24	O	**Write Clock Pattern:** Active high. This signal provides the write clock pattern for external logic, like the SCB68459 DPLL, to use with WRDAT and WCLOCK to generate the write data pulse stream to a disk unit. WRCLK is an NRZ signal which changes state on the rising edge of WCLOCK.
TICKLER	17	19	O	**Tickler:** Active high. This signal is used to control an external PLL. When TICKLER is asserted, the external PLL should output the crystal controlled clock to the WCLOCK/ RCLOCK input. With TICKLER is low, the PLL should synchronize on the incoming disk data and generate read clock to the WCLOCK/RCLOCK input.
SECTOR/ INDEX	31	34	I	**Sector/Index:** Active high. The IMDC uses the rising edge of the signal to generate the read/ write sequence. Hard sectored disk drives output a pulse for each sector boundary for the IMDC. Soft sectored disk drives have a once around index pulse to define the start of a track.
LOCAL	10	11	O	**Local:** Active high. During LOCAL mode, this line controls the output enable on the bidirectional buffers of the address/data lines.
OWNN	4	5	O	**Own:** Active low. This output is asserted by the IMDC during the DMA mode to indicate bus mastership. It can be used to enable external address/data and control buffers. Inactive in REG and IDLE modes.
DDIR	9	10	O	**Data Direction:** Active high. This signal is active during the DMA and REG modes. It controls the direction of the data through the bidirectional buffers on the address/data bus. DDIR is asserted during a read operation of the IMDC.
V_{CC}	48	52	I	**+5 volt ± 5% power input.**
V_{SS}	29	32	I	**Power ground input.**

REGISTERS

Register Map

The IMDC is a memory transfer oriented device with minimal information transferred to the registers. The internal accessible register organization of the IMDC is shown in table 1. Register bit formats are shown in table 2. Each is 8-bits wide to allow interface to either 8 or 16-bit host systems. When the IMDC is interfaced to an eight bit system, A0 is tied to LDSN and the data strobe (DSN) is tied to UDSN.

Interrupt Source Register (ISR)

These bits are used to indicate which drive was the source of a command completion interrupt. Bit 4 reflects drive 0 as the source and bit 7 is for drive 3. The assertion of RESN will initialize all four bits to zero. The IMDC will not initiate the next command, if pending, until the host resets the interrupt source bit. The host should also reset the appropriate busy bit in the status and configuration register.

Drive Status and Configuration Register (DSCR)

[0]8/16 Bit Mode

This bit sets the length of memory and register data transfers. Byte transfers are initiated when bit 0 is set to zero. This bit is set to zero when RESN is asserted.

[7:4]Drive Busy

These bits are used by the host system to initiate an IMDC command operation for a particular drive. Bit 4 is used for drive 0 and bit 7 for drive 3. After the host system has set a busy bit to activate a drive, the IMDC responds by accessing the corresponding event control area (ECA) and performing the requested action. The host system can abort the current drive operation by resetting the busy bit before the operation is completed. The assertion of RESN will initialize the busy bits to zero (only time the IMDC writes into this register).

Event Control Area Pointer Registers (EPH/EPMH/EPML/EPL)

These four registers are used by the host system to direct the IMDC to a table of pointers. The table consists of four 2 word addresses that point to the location of the four ECA blocks. The pointers are arranged in

Preliminary

Table 1. IMDC ADDRESS MAP

ADDRESS BITS[1]	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET
2 1 0				
0 0 0	EPH	ECA pointer high	R/W	Yes
0 0 1	EPMH	ECA pointer middle high	R/W	Yes
0 1 0	EPML	ECA pointer middle low	R/W	Yes
0 1 1	EPL	ECA pointer low	R/W	Yes
1 0 0	IVR	Interrupt vector register	R/W	Yes
1 0 1	ISR	Interrupt source register	R/W	Yes
1 1 0[2]	DSCR	Drive status and configuration register	R/W	Yes
1 1 1		Reserved		

NOTES:
1. A0 = 0 for UDSN asserted, A0 = 1 for LDSN asserted for 16- bit mode.
2. In 16-bit systems, the data for this register must be in data bits D0 – D7.

Table 2. REGISTER BIT FORMATS

INTERRUPT SOURCE REGISTER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DR3 INT	DR2 INT	DR1 INT	DR0 INT	<——— NOT USED ———>			

DRIVE STATUS AND CONFIGURATION REGISTER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
DR3 BUSY	DR2 BUSY	DR1 BUSY	DR0 BUSY	<——— NOT USED ———>			8/16 MODE

ascending order by drive number. EPL is the least significant byte and EPH the most significant byte. All four registers are cleared to zero by RESN asserted.

Interrupt Vector Register (IVR)

The IVR contains the value to be placed on the data bus upon receipt of an interrupt acknowledge from the CPU. The contents of this register are initialized to H'0F' by a reset.

OPERATION

The IMDC is an intelligent controller with on-chip microprogrammed CPU and interface circuitry. Two interfaces are provided, one to the host CPU and the other to the disk drive. The host interface contains a complete DMA interface compatible with the SCN68000 bus. After the IMDC accepts a command, the DMA interface handles all transfers between the system memory and the IMDC. The IMDC signifies through an interrupt signal that the command is completed.

Operation Initiation

The host must initialize the IMDC before it can process any command requests. The initialization information must be passed from the host to the IMDC control registers. The information consists of the data byte or word transfer mode selected in the drive status and configuration register. The host must also load an interrupt vector to the interrupt vector register and load the event control areas registers with the start location of the ECA pointer table in system memory.

A new command is requested using the drive status and configuration register. The user sets a bit corresponding to the drive to be serviced. This causes the IMDC to start execution of the command for the requested drive. This can be performed as long as the bus is available. The IMDC accepts the request but does not necessarily begin to process it. If the IMDC is currently doing disk data transfers to another disk drive, it will accept the request and treat it as a pending request for disk drive service. It does this in order to prevent forcing the host processor to wait for it to complete its current processing. The IMDC will execute parallel seek operations for floppy and SA1000 type disk drives.

Because four different drives can be on line simultaneously, the possibility exists of more than one pending request to appear in the drive status and configuration register. In this situation the IMDC will process the drive requests in ascending order (i.e., 0, 1, 2, and 3) and then start again with drive zero.

It is also the responsibility of the host to insure the drives that are to be controlled are in a ready state prior to issuing commands. Commands issued to drives that are not ready will result in the command being aborted. Also if a command is requested on a drive that goes from ready to not ready, the IMDC will abort the execution of the command. In both cases, an interrupt cycle will be initiated.

When RESN is asserted, the IMDC will go through an internal initialization program which clears the control registers and the interrupt vector will be set to H'0F'. All bus and control lines will be cleared and the IMDC will enter an idle loop.

DMA Operation

After a command is started, it is executed without further communication with the host system. All memory data transfers are handled automatically by the on chip DMA controller. The IMDC indicates that it wishes to become the bus master by asserting its bus request (BRN) output. The processor acknowledges the request by asserting its bus grant (BGN) output which puts the bus up for arbitration.

The IMDC will be the next bus master when its BGN input is asserted. The IMDC then waits for address strobe (ASN), data transfer acknowledge (DTACKN), and bus grant acknowledge (BGACKN) to become inactive. It then asserts the BGACKN output to become the bus master and negates the BRN output. The IMDC then proceeds with the transfer of data between itself and memory. After the data transfer phase, the IMDC relinquishes bus mastership by negating the BGACKN output. Flow charts for the transfer operations are shown in figures 1 and 2. Refer to the

timing section for the equivalent timing diagrams.

DMA Transfer Rates

The serial read/write I/O portion of the drive interface is a dedicated slave of the IMDC disk drive interface; that is, a serial string is sent or received by the interface without interruption. The DMA interface, however, must surrender the bus for arbitration after the requested number of operands have been transferred. If the number of transfers does not provide the IMDC with sufficient bus access time, it is conceivable that the DMA may fail to keep track with the disk I/O operation. This situation can occur if the bus is lost for a 'long' period of time. At the IMDC, the serial disk transfer rate is 1.6μsec per word at a 10MHz data rate. The IMDC processor operates using a 312.5nsec period clock. Three cycles or 937.5nsec are required to perform one DMA transfer. If the system bus is unavailable to the IMDC DMA for an average time greater than the difference between the drive and DMA transfer rates (662.5nsec), overflow or underflow of the FIFO buffer can result.

As an example, consider a Winchester disk transfer data of 5Mbits/second. If the transfer count is set to 16 words per transfer, the IMDC will transfer each 32 byte block in 15 microseconds. The transfer time will be 51.2 microseconds for the disk data to put 32 bytes into the FIFO. The difference between the two transfer block rates is 36.2 microseconds. This is the maximum time which the IMDC does not require the system bus.

To avoid this situation the DMA must not lose the bus for a period exceeding the difference in transfer rates. This can be accomplished in a variety of ways. One approach is to allow the DMA to have the bus mastership for the entire sector transfer. A second approach is delay the DMA until a sector has been read into the FIFO buffer and to delay writing to the disk until the FIFO is filled with one sector of

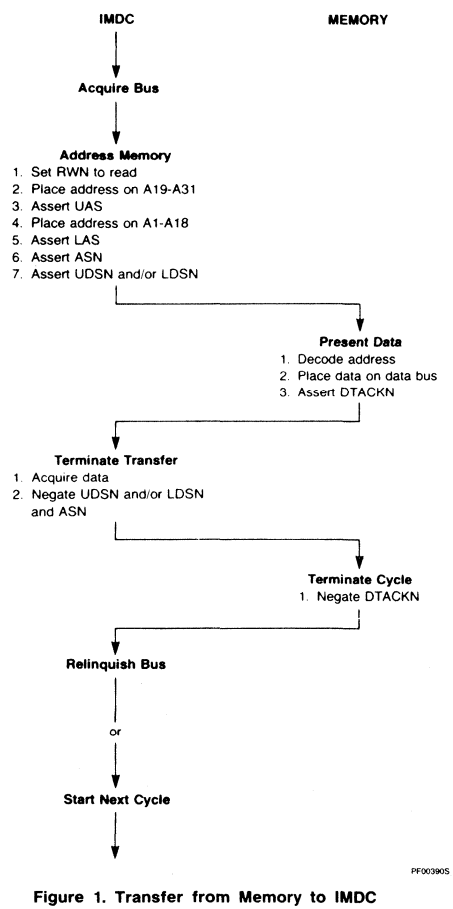


Figure 1. Transfer from Memory to IMDC

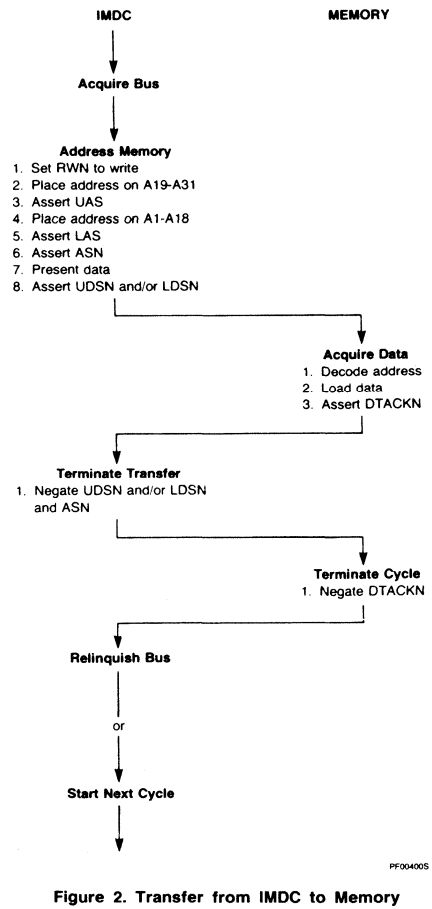


Figure 2. Transfer from IMDC to Memory

Preliminary

data. The first approach severely restricts the availability of the system bus to other devices.

The latter approach requires a minimum FIFO buffer equal to the size of a sector and introduces a delay into the IMDC operation.

As a result, both approaches place stringent limitations on the system and reduce the throughput of the device.

Another alternative is to assign the bus arbitration such that the IMDC always has highest priority and that the bus must be returned in a time segment shorter than the transfer rate difference. This latter solution requires external circuitry to implement and still does not guarantee the IMDC bus access time is sufficient. To provide the user with maximum system flexibility, without severely restricting the bus to other devices, the IMDC has been structured to allow the user to specify the number of operands that the DMA can be master of the bus. The ECA Command Option field permits the user to select the number of operands that can be transferred before the bus is returned for arbitration. This technique allows the user to customize the DMA operation to the needs of the system.

I/O DESCRIPTION

The disk interface consists of two sections. The input and output ports that sense and generate slow changing or static control signals and the serial read/write data section.

Table 3. INPUT PORT DEFINITION

BUS LINE	SIGNAL NAME	DEFINITION
D0	INDEX	Rigid disk index signal (hard sector mode) must be greater than 500ns
D1	READY	Drive ready
D2	SEEKC	Seek completed
D3	TR0A	Track zero first signal
D4	TR0B	Track zero second signal
D5	WFA	Write protection for floppies
D6	WFB	Write fault
D7	TWO	Two sided (signal valid for floppies only)
D8 – D15		Not used

Table 4. OUTPUT PORT DEFINITION

BUS LINE	SIGNAL NAME	DEFINITION
D0	HEAD1	Head select 2**0/side select for floppy disks set to '0' for single sided floppy disks
D1	HEAD2	Head select 2**1
D2	HEAD4	Head select 2**2 that can be transferred
D3	HEAD8	Head select 2**3
D4	HEAD16	Head select 2**4
D5		Not used
D6	STEP	Step pulse 10μsec long (period set in ECA)
D7	DIR	Direction of head movement (a high corresponds to head movement toward the disk spindle)
D8	LWC	Low write current
D9	MOT	Motor on
D10	PRECOM	Precompensation enable
D11	HDL	Head load for floppy disks
D12	SEL0	Drive 0 select
D13	SEL1	Drive 1 select
D14	SEL2	Drive 2 select
D15	SEL3	Drive 3 select

Input Port

The input port consists of eight lines to input disk status information to the IMDC. All signals are active high. To accommodate different drives, signals TROA, TROB, WFA and WFB are used. The IMDC will respond to any one of the signals which becomes active high. The drive will not activate these signal simultaneously.

Output Port

The output port consists of the signals corresponding to bus lines D0 through D15. All signals are positive logic. A practical implementation of the output port would consist of standard octal registers.

EVENT CONTROL AREAS

The host system communicates with the IMDC through the event control areas (ECAs) which reside in system memory. An ECA parameter block is set up for each disk drive (up to four) to be controlled. These areas contain information that is required by the IMDC to execute a disk command. This information includes data about the requested command and the disk drive. The host prepares for IMDC operation by loading the ECA pointer table in system memory with the address of each of the ECA blocks. The disk drive to ECA block assignment is determined by the relative position of the pointer in the table. The first table entry corresponds to drive zero, the second entry to drive one, etc. (see figure 3).

The IMDC microprogram will use the data contained in the ECA block to generate the disk interface signals and perform the requested drive I/O. Prior to informing the host of a command completion, the IMDC writes the return status information to the appropriate ECA block memory. Table 5 describes the format of the ECA block.

Preliminary

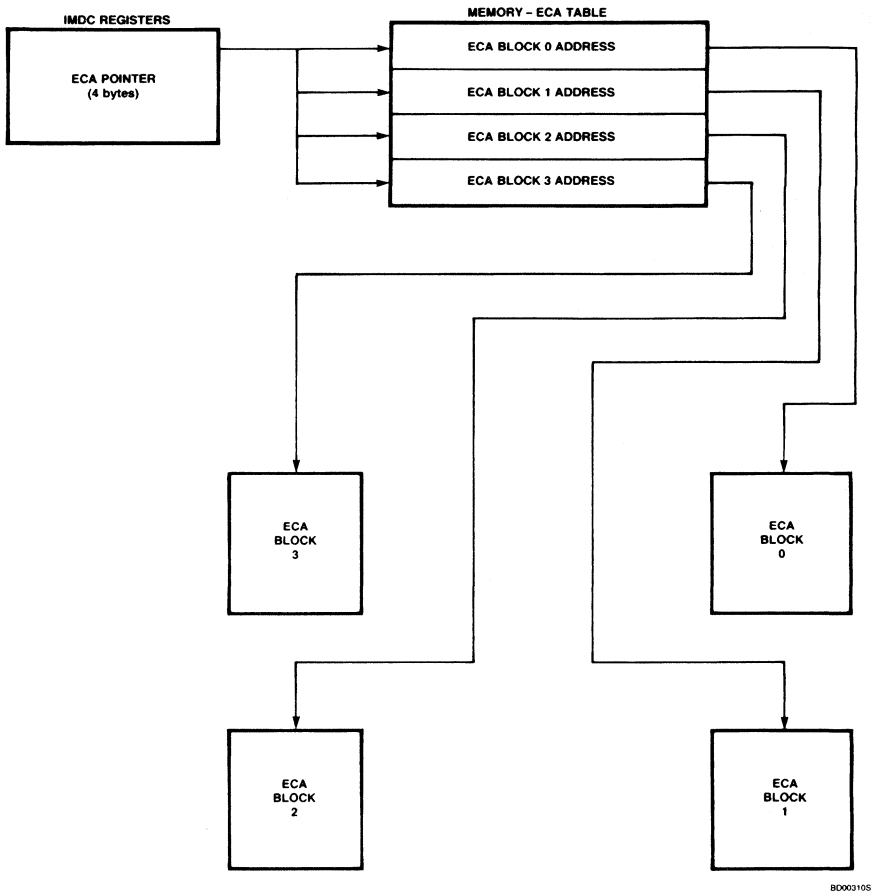


Figure 3. ECA Block Addressing

The communications between the IMDC and the host are established through the ECA. Alteration of the ECA by the host after a command has been accepted by the IMDC is not allowed.

The static and dynamic (see ECA fields) ECA command parameters should be verified by the host before new command execution is requested. During execution of a command involving multiple sectors, it is possible that dynamic values generated by the IMDC could be invalid. The host does not have access during this time to alter or even check these values, therefore, it is the responsibility of the host to not only guarantee the integrity of the initial parameters but also to insure that values calculated by the IMDC during the life of a command remain valid. For example, the IMDC cannot know that a calculated DMA address is valid until it attempts to transfer data.

Implied in the execution of any disk operation is reading of the ECA data by the IMDC to load the drive control parameters. At the conclusion of a disk operation, the IMDC performs a write operation to the corresponding ECA parameters to store the results of the command. Both reading and writing of the ECA fields by the IMDC utilize the host/DMA interface to arbitrate for the system bus and to perform the required handshaking.

ECA FIELDS

As an IMDC command proceeds through its execution; it references and alters various fields of the ECA. To gain a better understanding of the interactions between the IMDC and the various ECA fields, the ECA data can be separated into four general categories.

1. Command, command status, and execution parameters
2. Programmable record processing fields
3. Disk Format Fields
4. Drive Control Parameters

These categories differentiate the ECA fields not only in content but in terms of the access and alterability.

Preliminary

Table 5. ECA BLOCK FORMAT

WORD NO.	15 8	7 0	
00	Command code	Main status	
01	Extended status		
02	Max # of retries	Actual # of retries	
03	DMA count	Command options	
04	Buffer address most significant word		
05	Buffer address least significant word		
06	Buffer length request		
07	# of bytes transferred		
08	Cylinder number		*
09	Head number	Sector number	*
10	Current cylinder position		
11	PRP command control word		**
12	Location of SCWT, most significant word		**
13	Location of SCWT, least significant word		**
14	Scan terminator	Reserved	**
15	Maximum record length – 1		**
16	N0 – Pre index gap	N1 – Post index gap	***
17	N2 – Sync byte count	N3 – Post ID gap	***
18	N4 – Post data gap	N5 – Address mark cnt	***
19	Reserved		***
20 – 22	ECC mask (3 words)		
23	Motor on delay	# of heads	
24	Ending sector #	Stepping rate	
25	Head setting time	Head load time	
26	Seek type	Phase count	
27	Low write Current boundary track		
28	Precompensation boundary track		
29 – 31	ECC remainder (3 words)		****
32	Maximum number of cylinders per surface		
33	Sector length/first sector	Flag byte	
34 – 35	B – tree pointer (2 words)		**
36 – 45	IMDC working area 10 words		

* Physical starting sector number.
** Programmable record processing parameters.
*** Track format fields.
**** ECC remainder will be aligned to the MSB byte of this field.

Command, Command Status and Execution Parameters

These parameters consist of the ECA data fields required by the IMDC to execute a command. The fields can be further separated into static fields and dynamic fields. The static fields are valid for the duration of a command; the IMDC does not alter any information in these fields as long as the command is being processed. The dynamic fields provide the 'local storage' needed to execute multiple sector commands. The IMDC uses the dynamic fields of the ECA to maintain the current execution status of the on-line disk drives.

Command Code (1 Byte)
The command code indicates the command to be executed. Table 6 lists the valid IMDC commands and their associated codes. For a complete description of each command refer to the command description in the Command section.

Main Status (1 Byte)
This field contains the general oriented status information about the command execution. This encoded byte is updated at the completion of the current command and before the interrupt is asserted. Table 7 shows the possible values which may be returned by the IMDC.

Extended Status (2 Bytes)
This field contains specific bit oriented status information about the command execution. If errors result during the execution, the corresponding bit will be ORed into extended status and kept there. This OR function can be used by the host system for error logging. The extended status will be reset to zero by the IMDC at the beginning of the command execution. The error definitions are shown in table 8.

Preliminary

Table 6. COMMAND CODES

COMMAND MNEMONIC	HEX CODE	DESCRIPTION
WRMS	00	Write multiple sector
WRDD	01	Write with deleted data flag
VER	10	Verify
REMS	11	Read multiple sector
PRP	12	Programmable record processing
TSR	13	Transparent sector read
RETD	20	Read identifier
FORM	40	Format
CALB	41	Recalibrate to track zero
CORR	81	Correct data
DIAG	80	Diagnostic

Table 7. MAIN STATUS CODES

DECIMAL VALUE	DESCRIPTION
0	Correct execution without error
1	Irrecoverable error which cannot be completed (auto retries are attempted, see extended status)
2	Drive not ready
3	PRP operation unsuccessful
4 – 5	Not used
6	Command rejected
7 – 9	Not used
10	Command abort (busy bit reset by host)
11 – 255	Not used

Table 8. EXTENDED STATUS

DATA BIT	DESCRIPTION
0	Write fault
1	CRC/ECC error on data or ID
2	FIFO overrun/underrun
3	No identifier found
4	Not used
5	Deleted data address mark
6	Write on write protected diskette
7	Positioning error
8	Data part timeout
9	Not used
10	Uncorrectable data error (ECC)
11	Not used
12	Not used
13	Positioning timeout
14	Not used
15	Bus Error Fault (DMA operation)

Maximum Number of Retries (1 Byte)
This parameter specifies the maximum number of retries per command (not per sector) that the IMDC attempts, after a disk operation error. Loading a zero value indicates that no retries should be attempted. This byte will not be used if a bus error or FIFO under/overrun error occurs during the command execution. These two errors will cause an immediate command abort. If an identifier is not found, 16 revolutions of the disk will be tried before exit.

Actual Number of Retries (1 Byte)
This byte is set by the IMDC to the actual number of retries executed per command. All retries are accumulated by the IMDC on a sector by sector basis.

DMA Count (1 Byte)
This byte contains a DMA transfer count. If a zero value is specified, the IMDC transfers only one operand and surrenders the bus. A transfer amount of 1 to 16 can be specified.

Command Options (1 Byte)
This byte contains options that are to apply to the current command. Valid options for handling of deleted-data address marks are described in table 9. Sectors with the deleted-data address mark (bits 0, 1 of the option field) will be handled as shown for read operations, excluding those associated with the PRP command. Some retries may be necessary in these steps. They all start with a new identifier search.

Buffer Address
This is a 31-bit starting buffer address for the DMA transfers. The LSB of the address field defines the even or odd address of the 16 bit words when in 16-bit mode. For the 68000, only the first 23 bits are used for addressing. The additional bits can be used to implement the function codes. It should be noted that the IMDC does not protect the upper address lines, AD25 – AD31, from counter overflow. It is up to the operator to prevent transfers between the IMDC and memory greater than the 68000 addressing range. For PRP commands, this field contains the pointer to the content of the matched record data (see Programmable Record Processing section).

Buffer Length Requested (2 Bytes)
This field contains the requested number of bytes to be transferred. This field implicitly defines the number of multiple sector transfers to be performed. A seek only will occur when a zero length is specified. For the format track command, this value is used to terminate the operation.

Preliminary

Table 9. COMMAND OPTIONS

BIT	FORMAT	OPTION DESCRIPTION
1 – 0	00	FM, IBM single density format
	01	MFM, IBM double density format
	10	Programmable disk format, 1 byte cylinder
	11	Programmable disk format, 2 byte cylinder
2		Not used
4 – 3	00	Select CRC-CITT (CRC-16) polynomial
	01	Select 32 bit ECC polynomial
	10	Select 40 bit ECC polynomial
	11	Not used
5	0	The sectors with deleted-data address mark will be skipped as if it did not exist. A successful CRC/ECC check is not required.
	1	The data of the sector with deleted-data address mark will be transferred to the host system and the operation terminated
6		Not used
7		Hard sectored disk

Actual Number of Bytes Transferred (2 Bytes)
This field indicates how many bytes have been actually read or written during a command execution. The value can be used, for example, for a location of a problem sector with an irrecoverable error. The value of this field can be changed by the IMDC during the execution of a command and used as an intermediate field.

Physical Starting Sector Number (4 Bytes)
Cylinder, head, sector – This field contains the physical disk sector location at which the command is to begin execution. The format of the field is shown below:

Cylinder High	Cylinder Low
Head Number	Sector Number

Current Cylinder (2 Bytes)
These two bytes will be updated by the IMDC after each positioning. At the beginning of the IMDC reset, they should be set by the host system to zero, because most of the hard disks have built-in automatic recalibration.

Programmable Record Processing Fields (10 Bytes)

The programmable record processing fields are described in the Programmable Record Processing section.

Disk Track Format Fields (8 Bytes)

The disk track format fields are described in the Disk Track Format section.

ECC Mask (6 Bytes)
These six bytes define the error correction polynomial. The standard notation of a polynomial is: $X^{40} + k_{39} X^{39} + \bullet\bullet\bullet + k_1 X' + k_0 X^0$

The coefficients k_{39} through k_0 may be any combination of 1s and 0s. For a given polynomial, an equal combination of bits in the mask register will initialize the IMDC logic to generate or check the same polynomial on the disk.

Drive Control Parameters

These fields contain information that define the disk interface to the IMDC. This information is provided by the host system and is not altered by the IMDC. The IMDC microprogram, that controls the disk drive I/O issued, continually references this information during the execution of a command. Because of the potentially large number of possible variations of interfaces, the IMDC cannot provide extensive verification of the parameters contained in these fields. Protection and verification of these fields are primarily the responsibility of the a host operating system and not the IMDC.

Motor on Delay (1 Byte)
This field contains motor on delay (period) in one second units.

Number of Heads (1 Byte)
This value is the number of heads on the disk unit. The maximum value is 128.

Ending Sector Number (1 Byte)
This value is the last sector on the track (cylinder) for the disk unit.

Stepping Rate (1 Byte)
This field contains head stepping rate (period) in 500μsec units if applicable.

Head Settling Time (1 Byte)
This field contains head settling time in 500μsec units. If a non-zero head settling time is specified, the IMDC assumes that a seek complete is not available from the drive.

Head Load Time (1 Byte)
This field contains head load time in 500μsec units. For all hard disks, this value must always be set to zero by the host system.

Seek Type (1 Byte)
This field defines what type of seek positioning is to be performed.

VALUE	DESCRIPTION
0	Normal single step seek
1	ST506 with accelerated seek
2	Disk with buffered seek

Phase Counter (1 Byte)
This field contains the phase counter which is a status of the command execution set by the IMDC.

Low Write Current Active Track (2 Bytes)
This field defines, to the IMDC, the track at which low write current (LWC) output signal, on the output port (bit 8), is to be asserted.

Precompensation Active Track (2 Bytes)
This field defines to the IMDC the track at which precompensation signal (PRECOM) is to be asserted at the output port (bit 10).

ECC Remainder (6 Bytes)
This field contains the ECC remainder generated by the IMDC read operation. This returned value will be zero unless an error is detected.

Maximum Number of Cylinders (2 Bytes)
These bytes define, to the IMDC, the maximum cylinder count for the disk unit on line. If an operation is specified beyond this boundary, the IMDC will abort the command.

Sector Length (1 Byte)
This byte defines the sector length as a power of two multiple of 128 bytes, i.e., sector length = $(2^{\#})$ * (128 bytes). The minimum sector length of 128 bytes is specified by a zero value while the maximum length of 4096 is specified by 5.

Flag Byte (1 Byte)
This byte is placed in the ECA by the IMDC during a transparent sector read (TSR) command operation. It is the flag byte for the data field part of the sector.

B-Tree Pointer (4 Bytes)
The IMDC loads the physical sector number of the last record match when in the PRP command mode and in the B-tree scan function (see the Programmable Record Processing section for further details).

Preliminary

IMDC Work Area (20 Bytes)
These ten words are reserved for the IMDC microprogram. The meaning of certain bytes of this area will be determined by the command being executed.

COMMANDS

The following is a generalized description of the sequence performed by the IMDC to execute a command:

a. The IMDC checks if the drive is available. If it is not, the IMDC generates an error status and a completion interrupt.
b. The IMDC executes a track seek if necessary. For drives which have a seek complete signal (as indicated by a zero value in the head settling time field of the ECA), a command termination can be caused by the timeout of the drive signal SEEK COMPLETE. Drives for which the head settling time is non-zero, the IMDC waits the specified time after issuing the stepping pulses (that is, there is no timeout on the seeks).
c. The head will be selected, or for a floppy disk, loaded. The IMDC will wait four byte times to insure head switching has occurred.
d. The IMDC reads the sector identifier to locate the requested sector. If the IMDC is unable to perform the sector identifier read, a retry is attempted for as many times as specified in the ECA maximum number of retries parameter. If the retries are all unsuccessful or if no retries are specified, the command is terminated (step g) and the status 'no identifier' is returned. If the matched sector identifier indicates a bad sector (see Track Format description), the IMDC uses the replacement information to perform steps b and c for the new sector. This bad sector replacement is not counted as a retry.
e. After the requested sector has been located, the IMDC performs the sector disk I/O (read or write) and DMA operations. If the operation cannot be completed, the retry processing described in step d is attempted. A failure in this portion of the command is indicated by the return status 'data part time out', which distinguishes it from the read identifier failure of step d.
f. If multiple sectors have been specified in the command, the IMDC assumes contiguous physical sectors: steps b, c, d and e are repeated for each sector. The IMDC automatically performs any track seek required if the next physical sector is located on the next cylinder. When the sectors are interleaved, the IMDC repeats step d until the correct sector is located.
   In the case of a bad sector replacement, the IMDC continues the multiple sector operation, after the replacement sector, at the sector physically located after the bad sector. It performs all the necessary track repositioning required to return the drive head to the next physical sector.
g. Upon command completion, all relevant ECA fields are updated and the appropriate bits of the 'interrupt state status' and 'drive status and configuration' registers are set and an interrupt signal is generated by the IMDC. Upon interrupt acknowledge, the IMDC presents the interrupt vector from the interrupt vector register on the data bus.

Command Description

In the command descriptions, the term FIFO is used to refer to the internal memory of the IMDC when it is used as a FIFO buffer.

Write Multiple Sector with Implied Seek (WRMS)
After the desired sector is located, the IMDC will write the complete data part; preamble, address mark, flag, data, CRC or ECC and postamble. The precise format is given by the drive type. The FIFO will be continuously filled with the new data from the buffer. The number of data bytes written in one sector is determined by the IMDC using information from the ECA fields. This operation is repeated until the number of sectors, implied by the buffer length field of the ECA, have been written.

Write with Deleted Word (WRDD)
This command differs from the normal write only in that the flag or address mark written is the deleted data address mark or flag.

Verify (VER)
Verify with implied seek is the same functionally as the read command, except the IMDC compares the ECC or the CRC for accuracy. No data is transferred to the system.

Read Multiple Sector with Implied Seek (REMS)
After a successful seek to the desired sector, the IMDC will start to read the data of the sector, fill the FIFO buffer and check the CRC or ECC code. This sector data is transferred through the DMA interface to the host system memory. This procedure is repeated until the number of sectors, implied by the buffer length field of the ECA, have been read. For sectors with the deleted data address mark, the IMDC will process them according to the options selected in the command option field of the ECA.

Programmable Record Processing (PRP)
Refer to the Programmable Record Processing section.

Transparent Sector Read (TSR)
This command will read a single error sector and transfer the data to the FIFO buffer regardless of a CRC or ECC check. Only the extended status will be correspondingly filled with all errors encountered. The transparent sector read can be used for diagnostic purposes and, with some manual help, for recovery of damaged data. In case of an incorrect CRC/ECC remainder, the IMDC discards the remainder into four bytes reserved in the ECA for this value.

Read Identifier (RETD)
This command will read identifiers as they come from the disk and fill the whole buffer with records consisting of:
- flag byte
- ID data
- CRC
- remainder generated by IMDC

A CRC error will not terminate the execution of the command. For soft sectored formats, the command execution will begin and end with the detection of an index pulse. For the hard sectored formats, the IMDC will detect and count sector pulses to determine the command termination.

Format a Track with Implied Seek (FORM)
The IMDC supports four different media formats (see Disk Track Format section). This command allows the user to write the format information, as specified by the ECA track format fields, to the recording media.

Recalibrate to Track Zero (CALB)
The function of this command is to retract the heads to track 0. The IMDC issues one step 'IN' and then steps 'OUT' until the signal track 0 becomes active or the maximum number of steps equal the number of cylinders from the ECA. The check on seek completed timeout will be made for the hard disk. The function recalibrate can be initiated automatically by two conditions:
- Encountered error in cylinder number by reading or writing.
- Writing zero into the current cylinder in ECA.

Correct (CORR)
This command provides an error mask which is used to correct the data in the memory. It uses the ECC remainder field of the ECA. The IMDC will put the error correction vector and its relative position from the end of the data buffer into the ECA.

The 24-bit ECC correction mask will be placed in the flag byte and first half of the B-tree pointer of the ECA block. This is the least significant byte of the 33rd word and the 34th word of the ECA block in memory as shown in table 5. The offset count will be placed in the second half of the B-tree pointer or the 35th word of the ECA block. This is the relative offset from the first byte transferred to memory.

Preliminary

Chip Diagnostic (DIAG)
The IMDC will exercise the 16-bit counters; the record counter, the record length counter, the buffer length counter, and the time out counter. If everything is functioning properly, the IMDC places a zero byte into the main status. Otherwise, an irrecoverable error bit value is set in the main status.

DISK TRACK FORMAT

The IMDC supports four different track formats:

- IBM 3740 single density
- IBM System 34 double density
- Programmable soft sectored
- Programmable hard sectored

In each of the formats, data on the physical media is separated into blocks of information or sectors. The format serves several purposes in this arrangement; it defines the structure of the sector, the location of the actual data, and provides gaps and sync bytes that allow an interval for any switching required by the drive hardware. These intervals, in turn, allow the IMDC to compensate for any variations in either the recording media and/or the drive hardware. Although each of these formats is well defined, variations of parameter values within a given format require that the IMDC provide the user with the capability to program them.

Because there is no one standard which defines track format parameters, a description of them and other pertinent definitions are included in this section. See table 10 for track format definitions. Tables 11 and 12 provide a summary of the formats and the programmable values. The following definitions apply:

The format command operation is performed by the IMDC as integral operations on a per track basis, as opposed to normal disk I/O which is on a sector basis. This technique was selected as a compromise to satisfy two conflicting – requirements command efficiency versus equal access of on-line drives to the IMDC resources. Obviously, these commands would be most efficient if allowed to monopolize the IMDC resources. However, this situation would prevent any other drive from being serviced until the command had been completed. Normally these commands are background tasks with the other disk operations having a higher priority. For this reason, allowing the IMDC to concentrate completely on either command is not a good system practice.

The IMDC becomes available for other processing after a full track has been processed. In the worst case situation for the format command, the time the IMDC will not be able to process the other drives will not exceed two disk revolutions. This would occur for disks in which the index pulse was just missed and a complete revolution is required to find the pulse.

The programmable soft sectored disk format, which is used mainly for hard disk drives, is a MFM format which is nearly identical to the IBM double density format. The hard sectored disk format is used for rigid disks with an internal sector clock and is similar to the soft sectored format, except that each sector starts with the sector pulse rather than a related byte count from the index pulse.

The format track command allows the user to write formatting information to the recording media. Specification of format parameters is accomplished by changing appropriate ECA fields. This structure gives the user a tremendous flexibility to accommodate, through changes in the ECA, variations that occur within a given format.

For non-hard sectored disk formats, the IMDC writes the sector identifier and fills the data part with the fill byte starting with the leading edge of the first index pulse and ending with the leading edge of the next index pulse. For hard sectored disk formats, the format information is written between sector pulses.

Tables 11 and 12 show the possible layout of the track information in table format. Table 11 is for the floppy parameters and table 12 is for the rigid disk format parameters. In table 11 each of the different fields on each track is described. Associated with each field is where the IMDC gets the data, either from the ECA register locations or through the DMA process from the system memory.

ECA Track Format Fields

For the four formats supported by the IMDC, eight parameters are required to specify the format of the recording media to the IMDC. The format parameters are programmed by changing the values of the appropriate ECA fields. The layout of the track format portion of the ECA is given in table 13.

N0 and N1 are ignored by the hard sectored format and only N1 is applicable to the programmable soft sectored format. N5 contains the number of address marks contained in the address mark subfields for the active format; valid values are either one, two or three. N2 must be at least four.

Format Parameters
Although exhibiting some differences, the parameters that constitute each of the four formats supported by the IMDC are basically similar. In each format the data on the disk-

Table 10. TRACK FORMAT DEFINITIONS

NAME	SUBFIELD	FORMATS	DESCRIPTION
N0	Index	IBM only	Pre-index Gap. This gap represents the number of bytes that appear prior to the index pulse.
N1	Index	All except hard sect	Index Gap. This gap represents the number of bytes that appear after the index pulse and prior to the ID subfield.
N2	ID, data	All	Preamble count or sync. This is the number of index sync bytes that precede the address mark.
N3	ID	All	Post ID gap. This count is the number of bytes that separate the ID subfield from the data subfield.
N4	Data	All	Post data gap. This count is the number of bytes that separate the data subfield to the beginning of the ID subfield of the next sector.
N5	ID, data	IBM	Address Mark Count. This contains the number of index address marks contained by the subfields. For single density formats, the count is one and for double density the count is three.
	ID, data	Prog	The number of data part address marks is a 1, 2 or 3. The number of ID address marks is always 1.

Preliminary

Table 11. SUMMARY OF FLOPPY FORMAT PARAMETERS

DESCRIPTION	IMDC USES	HEX DATA VALUE	FM CNT	HEX DATA VALUE	MFM CNT	
Pre-index gap	N0	FF	40	4E	80	
Sync field	N2	00	6	00	12	
Index mark	N5	FC/D7[1]	1	C2[2]	3	
Index flag		–	–	FC	1	
Index gap	N1	FF	26	4E	50	
Sync field	N2	00	6	00	12	▲
ID address mark		FE/C7[1]	1	A1[3]	3	
ID address flag		–	–	FE	1	
Cylinder	DMA		1		1	
Side	DMA		1		1	
Sector	DMA		1		1	
Record length	DMA	01[4]	1	01[4]	1	
CRC-CCITT			2		2	Repeat as
Post ID gap	N3	FF	11	4E	22	required
Sync field	N2	00	6	00	12	
Data address mark	N5	FB/C7[1]	1	A1[3]	3	
Data address flag		–	–	FB	1	
DATA (see note 4)		Fill byte	256	Fill byte	256	
CRC-CCITT			2		2	
Post data gap	N4	FF	27	4E	54	▼
Inter-record gap[5]		FF	170	4E	598	

NOTES:
1. Shows data pattern and clock pattern (clock pattern normally FF).
2. Shows data pattern; clock pattern should suppress clock bit between data bit 3 and 4.
3. Shows data pattern; clock pattern should suppress clock bit between data bit 4 and 5.
4. This example is for a 256 byte sector, others will have different values.
5. This is an approximate count.

Table 12. SUMMARY OF RIGID FORMAT PARAMETERS

DESCRIPTION	IMDC USES	HARD SECTOR		SOFT SECTOR		
		Hex Data Value	Pgm Cnt	Hex Data Value	Pgm Cnt	
Index gap	N1		22	4E	22	
Sync field	N2	00	13	00	13	▲
ID address mark	N5	A1[1]	1	A1[1]	1	
ID address flag		FE	1	FE	1	
Cylinder	DMA		1or2		1or2	
Head	DMA		1		1	
Sector	DMA		1		1	
CRC-CCITT			2		2	
Post ID gap	N3	00	3	00	3	Repeat as
Sync field	N2	00	13	00	13	required
Data address mark	N5	A1[1]	1	A1[1]	1	
Data address flag		FB	1	FB	1	
DATA (note 2)		Fill Byte	256	Fill Byte	256	
CRC-CCITT		note 3	2,4,6	note 3	2,4,6	
Post data gap	N4	00	3	00	3	
Inter-sector gap		4E	15	4E	15	▼
Inter-rec gap[4]				4E	346	

NOTES:
1. Shows data pattern; clock pattern should suppress clock bit between data bit 4 and 5.
2. This example is for a 256 byte sector; others will have different values.
3. Can be either a CRC-CCITT, 32-Bit ECC, or 40-Bit ECC Field.
4. Approximate count.

Preliminary

ette or disk is separated into a logical data block or sector. The sector becomes the smallest block of information that can be addressed directly by the IMDC.

The programmable soft sectored and the two IBM formats organize the physical disk into a circular path or track, which, in turn, is separated into several sectors. In this scheme, tracks on the media are referenced with respect to a physical index mark. An index mark pulse is generated by the media to indicate the beginning of a track. By specification of a track and sector, the location of a sector on the media is uniquely addressed (for at least one side of the media). The programmable hard sectored format also uses this track and sector structure, however, it differs from the other formats in that in addition to the index pulse, each sector is preceded by a sector pulse.

In each of the formats, a sector can be further separated into two parts or subfields – an ID and a data subfield. The ID subfield contains a unique identifier (or address) and description of the sector. The data subfield contains the actual data of the sector. Both subfields contain three types of information:

1. Address mark – unique character which precedes the data in the subfield. For the MFM formats, (non IBM single density) it is followed by a single character, the address mark flag, which further describes the subsequent data.
2. Data (either actual or about the sector).
3. Error report – pattern generated by the IMDC that is used to verify the data transmitted.

Both fields contain sequences of characters called 'gaps' and 'syncs' that are used to differentiate the sector subfields and to provide an interval that allows any hardware switching to be performed. As a result, these sequences provide any timing compensation due to variations in either the recording media or the disk drive.

For the programmable hard sectored and IBM formats, (formats that use the index pulse), a third subfield, the index subfield, is utilized. This subfield appears prior to the first physical sector of a track on the recording media and has subfields which contain gap and sync sequences. Unlike the other two fields it occurs only once per track and contains only the address mark information.

Table 13. ECA TRACK FORMAT

Word	15 — 8	7 — 0
16	N0 – Pre Index Gap	N1 – Post Index Gap
17	N2 – Sync Byte Cnt	N3 – Post ID Gap
18	N4 – Post Data Gap	N5 – Address Mark Cnt

Format Table

Table 14 contains the tables used by the IMDC during the format operation. Each table section is unique for the four different format types the IMDC can execute. The table is addressed by the SCWT pointer in the ECA block in RAM.

Provisions for a Bad Sector Substitution by Formatting

The IMDC command provides a convenient mechanism for handling bad sectors on the recording media. For example, consider the identifier (ID subfield) layout for a good sector using the programmable soft sectored format.

	Bytes	Coding Data/Clock
Preamble(sync)	N2	00
Address mark	1	A1/A0
Flag	1	FE (normal)
Cylinder number	1 or 2	
Head number	1	
Sector number	1	
Sector length	1	
CRC-CCITT	2	
Postamble	3	
Post ID gap	N3	

The data part of the sector is:

	Bytes	Coding Data/Clock
Preamble(sync)	N2	
Address mark	N5	A1/A0
Flag	1	FB (normal) or F8 (deleted)
Data	128X**L	L is the number between 0 and 5.
ECC	N6	
Postamble	3	
Post data gap	N4	

The media sectors with defects cannot be used for recording data. The host can replace any bad sectors it encounters during formatting with good sectors. Sector replacement is accomplished via the coding of identifiers (ID subfield) during the formatting. A bad sector is identified by the host placing an X'FF' in the sector length field of the ID part of the sector. The identifier for the bad sector is then extended with information pointing to the replacement sector.

Usually, there will be more than one identical bad sector identifier on the track for the same media defect. In this way, a later correct reading will be possible independent of the position of the media defect. During the read and write, the IMDC will automatically issue a seek to the replacement sector cylinder. There is no restriction on the placement of the substituting sectors. The identifier for a bad sector for the previous example would then appear as:

	Bytes	Coding Data/Clock
Preamble	N1	00
Address mark	1	A1/A0
Flag	1	FE (normal)
Cylinder number	1 or 2	
Head number	1	MSB set to 1 for bad sector
Sector number	1	
Cylinder number	2	
Head number	1	
Sector number	1	
CRC-CCITT	2	
Postamble	4	
Post ID gap	N3	

Preliminary

Table 14. FORMATS

IBM FM	
15	0
01	00
FE	00
80	N3-3
FF	Sector length
01	00
FB	00
Fill byte	00
FF	N4-5
FF	00
01	N5-1
DC	00
FF	00

IBM MFM	
15	0
00	02
A1	FE
80	N3-3
4E	Sector length
00	02
A1	FB
Fill byte	00
4E	N4-5
4E	00
00	N5-1
C2	FC
4E	00

PROG 1 AND 2 CYL	
15	0
00	00
A1	FE
00	N3-3
00	Sector length
00	N5-1
A1	FB
Fill byte	00
00	N4-2
4E	00
4E	00
00	00
00	00

Note that these tables are aligned to an even address boundary.

PROGRAMMABLE RECORD PROCESSING

The following definitions apply:

Key
Character string that the IMDC is to locate and match.

Scan
Search operation performed by the IMDC in trying to match the key.

Field (data item)
Smallest unit of named data. It consists of a string of characters that has a user defined significance. Records are formed by joining several fields together.

Record
Named collection of data items (fields) that has significance to the user.

Key Field
Field that contains the key data.

File
Named collection of all occurrences of given type of record.

The IMDC can be instructed to locate a specified string of characters within a logical data block on the recording media. After the string has been matched, the IMDC can perform the following additional functions:

- Retrieve and store the content of the logical data block
- Process a pointer from the data block
- Locate all data blocks that satisfy some search criteria

These primitive functions constitute the programmable record processing (PRP) capability of the IMDC, which can be used to form the basic support of more sophisticated applications such as:

- Directory processing
- Data block retrieval for data base management systems
- Processing of complex file structures (linked and mapped file structures)
- Searching of multilevel tree or network data structures

The programming for the PRP functions consists basically of two parts. One part involves loading of ECA fields with parameters that define the PRP operations and the physical representation of the data to be processed. The second part consists of table(s) which describe the search criteria on a character by character level.

PRP ECA Fields

The IMDC requires the following information to define the PRP operation:

1. PRP command control
2. Location of scan control word table
3. Location of matched record storage (not a separate PRP field; the IMDC uses the ECA buffer address field for this purpose)
4. Scan termination character
5. Maximum record length

PRP Command Control (1 Word)
This field contains the execution control parameters for the PRP command. The format of the command control field is:

B-tree scan (bit 1) — This parameter enables the B-tree scan function on the IMDC.

No data transfer — Returns pointer in ECA B-tree pointer.

Record type (bit 2) — This parameter contains a code which specifies the record type:

0 Fixed length. The record length is given in the maximum record length (MAXLEN) field of the ECA.

1 Variable length. The record is of variable length and uses the content of the scan terminator field of the ECA to determine the end of the record. If this terminator is not found, the IMDC uses the MAXLEN parameter to terminate the record search.

Location of Scan Control Word Table (2 Words)
This field contains the address of the start of the scan control word table in system memory.

Scan Terminator (1 Byte)
This character specifies the end of the scan field in the record.

Maximum Record Length (MAXLEN – 1 Word)
This field plus one is the maximum length of variable length records and is the length of the fixed length records.

Scan Control Word Table

The scan control word (SCW) table's primary task is to provide the character by character comparison template. It also is used to locate the key field and provide processing instructions for the IMDC as shown below. It must be located on an even word boundary.

15 0

Program length	Record start offset
Number of record to be scanned	
Scan control word(s)	

Preliminary

Bits 8 – 15
DATA – This 8 bit field is used in the following manner depending on the usage defined by the operation.
1. It contains the data to be matched against the key
2. It contains a count for repeating a previous instruction.
3. It contains the character used with the continue opcode.

Bit 5
SUB – This bit is used to indicate the last character of a subkey or the last character of the key to the IMDC. This flag is used by AZFF and SUCFF to control generation of the automatic initialization state for these devices. This bit must be set for the last byte of the key field, i.e., when bits 6,7 are 0,0.

Bit 4
ORENA – This bit is used to indicate to the PRP processor that the result of the previous subkey comparison is to be 'ORed' with the next logical comparison.

Bits 6, 7
OP – This 2 bit field is a command to the IMDC to perform one of the following operations independent of the resultant logical processing being performed.

7 6
0 0 Indicates the last byte of the key field.
0 1 Read this character (data) into system memory from this character until either:
1. Stop read command is issued, or
2. End of key, and no match has occurred.
1 0 Stop reading
1 1 No-op

Bits 0 – 3
OP CODE – This field is used to indicate either the logical comparison or some special control function is to be performed.

CODE	OPERATION
0	Repeat previous instruction (data + 1) times
3 – 1	Not used
4	SUCFF is unchanged
5	Continue scanning, no-op until 'data' byte is read
6	Not used
7	If ORFF is set, then store this character into the FIFO. If ORFF is not set, then use last point in FIFO as address of next sector to be scanned on the next level of the tree or plex.
8	Reset SUCFF if (CHAR GT Data) • (AZFF = 1).
9	Reset SUCFF if (CHAR GT or EQ Data) • (AZFF = 1).
10	Reset SUCFF if (CHAR LT Data) • (AZFF = 1).
11	Reset SUCFF if (CHAR LT or EQ Data) • (AZFF = 1).
12	Reset SUCFF if CHAR = Data.
13	Reset SUCFF if CHAR NEQ Data.
14, 15	SUCFF is unchanged

The number of scan control word tables required to describe a PRP function is determined by the data structure organization. For records organized in a relational structure, a single SCW table is required to provide the pertinent parameters. A complete description of the PRP operation is defined by the single SCW table and the ECA fields.

For records organized into a tree or plex structure, each level generally requires a new SCW table. Because each level may have a different record structure, certain ECA parameters describing the record structure must be updated (see PRP ECA fields discussion).

Note that a repeat operation cannot follow a continue operation.

Location of Matched Record Storage (2 Words)
The IMDC will use the previously defined 'buffer address' field.

PRP Status Flip Flops

To understand the operation of these flip flops, one must differentiate between the 'automatic initialization state' and the 'logical operation' of each of these internal devices. If this distinction is not made, some apparent conflicts in their logical state appears to exists. For example, a contradictory condition appears to occur for the accumulated zero flip flop (AZFF) for the last character of the subkey. According to the description of the flip flop, it is reset if the comparison fails. However, in the same description, it is stated that the AZFF is set if the character is the last of the subkey string. This apparent contradiction is easily explained; it is simply a problem of failure to recognize the sequence of events that is involved in execution of the 'logical operation' and the 'automatic initialization state'. The logical operations are performed by the flip flops prior to the automatic initialization state.

In order to help in making this distinction, the operation of each of the flip flops is described in terms of the logical operation and the automatic initialization state rather than in terms of the set/reset conditions.

Accumulated Zero Flip Flop (AZFF)
This flip flop is used to report the status of each character by character comparison. It is used to identify the first unsuccessful character by character match.

Logical Operation — The flip flop is reset on the first unequal character comparison.

Automatic Initialization State — The operation of the flip flop is initialized to the set condition:
1. At the beginning of a new record.
2. At the end of a subkey prior to the next subkey. The SUB bit of the SCWT indicates this condition.

Success Flip Flop (SUCFF)
This flip flop can be used to indicate the logical status of the subkey match operation. The AZFF cannot be used for this purpose since its primary task is to respond to the character by character search operation.

Logical operation — The SUCFF is reset by the interaction of the SCWT operation (OP code) and the status of the AZFF.

Automatic initialization state — This flip flop is set for the following conditions:
1. At the beginning of a new record.
2. At the end of a subkey if the ORENA bit of the SCWT is set.

OR Flip Flop (ORFF)
This flip flop is used to indicate the resultant scan status for the entire key field. The ORFF will always be in the set condition at the conclusion of a successful key scan, regardless of the logical operation performed between the subkeys.

Logical operation — The ORFF is set at the end of a successful subkey comparison; this condition is indicated by the SUCFF and the ORENA bit of the SCW table both being set.

Automatic initialization state — RFF is reset for the following conditions:
1. At the beginning of the first record.
2. At the beginning of all records to be scanned.

Programmable Record Processing Operation

Although the programmable record processing command utilizes most of the basic disk read sequence described earlier, its execu-

tion involves significantly more character processing than is performed for the basic read operation. For the PRP command, the function of the IMDC memory is split; one portion is allocated to store the PRP program contained in the SCWT table and the remaining portion is used as a FIFO buffer. The term 'FIFO' for this discussion of the PRP command refers to the memory available after the SCWT has been loaded. The command is 'normally' executed in the following sequence:

1. Transfer the SCW table data from system memory to a portion of the IMDC memory. The remainder of the IMDC memory is used as a FIFO to store the data from the record.
2. Perform disk positioning I/O to locate the drive read/write heads to the first sector at which the PRP is to operate.
3. Perform a read of the sector ID part to locate the data part.
4. Read an operand from the data part of the sector. The IMDC performs processing of this operand in order to locate the key field. This step is repeated until the key field is located.
5. After the key field is located, it is compared character by character with the SCWT template. This step is repeated until a match or no match with the key is determined.
6. If the scan does not produce a match the current record is skipped and the next record is located for processing by repeating steps 3, 4 and 5.

Programming Parameters Supplemental Information

The following describes programming parameters in more detail.

Record Format Types

Records to be processed by the IMDC, regardless of the overall data organization, can be formatted in one of two types.

1. Fixed – Records consist of a fixed number of characters. The user must pass a record length parameter value to the IMDC.
2. Variable – Records are of variable length. Requires the user to specify a unique termination character to identify the end of each record.

Since all records organized in the relational data structure are identical, the selected record format is applicable to all the records being processed. For data organized in a tree or plex structure, each level may have a different format type, therefore, each requires this parameter to be specified.

Key Field Location

The IMDC allows the user to specify the location of the key field in the record. The SCW allows the user to program the necessary information to locate the key field. The record format type determines how the key field is specified.

1. For fixed length records, the key start occurs at some fixed number of characters from the beginning of the record. For these records, the user must provide the IMDC with this count.
2. For variable length record, the key field location is specified by the number of data fields occurring prior to it in the record. Data fields consist of data that is bordered by a 'marker' character. Normally, this character will be some unique control character like a tab, line feed or return code. The PRP hardware will count the occurrence of the markers to locate the key field. Specification of the variable length field requires the user to provide the 'marker' as a parameter to the PRP.

Record Offset Start

The recording media often contains some header or identification information prior to the first record. The IMDC allows the user to specify where the PRP control is to become active. This allows the IMDC to skip any information that is not to be processed by the PRP. This parameter is referred to as the 'record start offset'.

Number of Records to Process

Because a user does not always know how many records are to be processed, the IMDC provides mechanisms to limit the number of records that are to be processed. First, the user is required to specify the 'number of records to process' parameter which is a count of the maximum number of records to be processed. In addition to this count, the user must also specify either an 'end of record' character for variable length fields or a record length count for fixed length records. The IMDC will monitor the number of records processed and will terminate the processing when the total equals the 'number of records to process', or if the 'buffer length' is exceeded.

Programming the PRP Feature

One of the most powerful aspects of the PRP feature is the capability it provides the user to effectively specify the search criteria on a character by character basis. The PRP feature not only allows the user to establish the criteria for success, but also to specify the action to be taken after completion of a search. The PRP program consists of two major parts. The first part essentially defines the basic requirements of the processing and the structure of the records; it is given once during initialization of the feature to the IMDC. It consists of loading several ECA fields dedicated to defining the parameters of the PRP operation to the IMDC.

Scan Control Word Table

The second part of the PRP program consists of a template which is used to perform the character by character match of the record data as it read from the media. The template consists of the character string data to be matched and processing instruction for each character of the string. The IMDC utilizes a 16-bit word to implement this comparison/instruction. It is loaded by the host into system memory and is transferred by the IMDC into its memory before execution of the PRP command. The SCWT is used in conjunction with three internal IMDC statuses to control and monitor the operation of the PRP search. Because of the interaction, they will be described prior to presentation of the table format.

PRP Key Status

To implement the PRP feature, the IMDC performs a character string search of the key field against the scan control word data. The IMDC generates and stores the status of this comparison in three flip flops dedicated to reporting the status of this operation. The logical operation of these flip flops is as follows:

For the most general case, the key field is composed of substrings. For convenience these substrings shall be referred to as subkeys for the remainder of this section. The desired logical result of the scanning of the entire key field is the logical combination of the constituent subkeys. For this general situation, there is the possibility of three levels of comparison statuses that occur in attempting to match a single key field – character, subfield, and entire key field each has a status that must be monitored by the IMDC. The IMDC has three flip flops that generate a status that allows these conditions to be monitored. Consider the example:
YYYYYY985DEF12580XXXXXXXXXXXX

This key field consists of five subkeys; YYYYYY, 985, DEF, 12580 and 'XXXXXXXXXXXXX', referenced arbitrarily as subkeys A, B, C, D, and E, respectively. Suppose that each subkey has the following logical condition associated with it.
A – Value is not processed by the PRP
B – Value greater than 980
C – Value equal to DEF
D – Value less than 20000
E – Value is not processed by the PRP

To perform the scan operation for this example requires that the IMDC be able to generate and monitor the condition of the character by character comparison (level 1), each subkey (level 2) and finally the entire key (level 3). The SCWT has been structured to allow specification of such combinations.

Preliminary

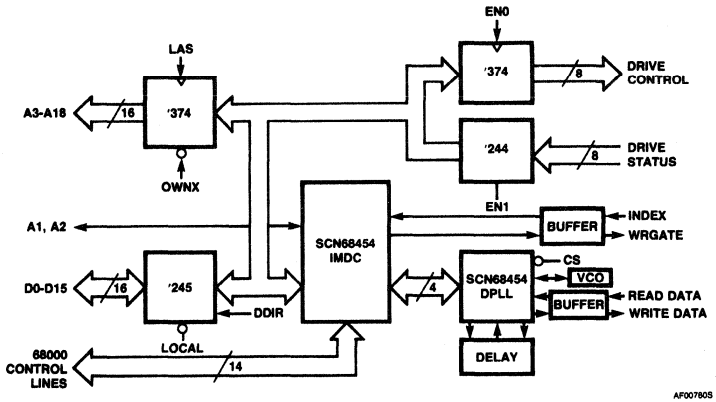


Figure 4. IMDC Application

Preliminary

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.3 to +7.0	V
Input voltage[2]	−0.3 to +7.0	V
Operating temperature range[3]	0 to +70	°C
Storage temperature	55 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5V ± 5%, V_{SS} = 0V; T_A = 0°C to +70°C[4,5]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
V_{IH}	Input high voltage		2.0	V_{CC}	V
V_{IL}	Input low voltage		GND − 0.75	0.8	V
I_{IN}	Input leakage current RERUNN, RESN, SCLK, IRQN, IACKN, BRN, BGN, BGACKN, EN0, EN1, UAS, LAS, REDAT, WRGATE, WCLOCK/RCLOCK, WRDAT, WRCLK, TICKLER, SECTOR/INBEN, LOCAL, OWNN, DDIR	5.25V		20	μA
I_{TSI}	Three-state (off state) input current A1, A2, D0 − D15, ASN, UDSN, LDSN, R/WN, DTACKN	2.4V/0.4V		20	μA
V_{OH}	Output high voltage A1, A2, D0 − 15, ASN, UDSN, LDSN, R/WN, DTACKN, EN0, EN1, UAS, LAS, WRGATE, WRDAT, WRCLK, TICKLER, LOCAL, OWNN, DDIR	I_{OH} = −400μA	2.4	V	
V_{OL}	Output low voltage A1, A2, D0 − D15, ASN, UDSN, LDSN, R/NN, DTACKN IRQN, BRN, BGACKN, EN0, EN1, UAS, LAS, WRGATE, WRDAT, WRCLK, TICKLER, LOCAL, OWNN, DDIR			0.5	V
P_D	Power dissipation			1.5	W
C_{IN}	Capacitance	V_{in} = 0V, T_A = 25°C f_o = 16MHz		10	pF

NOTES:

1. Stresses above those listed under absolute maximum rating may cause permanent damages to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltage larger than the rated maximum.
3. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8 and 2.0V as appropriate.

Preliminary

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5V ± 5%, V_{SS} = 0V; T_A = 0°C to +70°C[4,5] (see figures 5 – 18)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS		UNIT
			Min	Max	
1	5	Cycle time		62.5	ns
2	5	Clock pulse width low	21		ns
3	5	Clock pulse width high	21		ns
4	5	Rise time		10	ns
5	5	Fall time		10	ns
6	6	Reset pulse width	100		ns
7	7, 8	A1 and A2 set-up to CSN low	20		ns
8	7,8	Local low after CSN low		1	µs
9	7	D0 – D15 valid data from ASN, CSN, and UDSN or LDSN low		1	µs
10	7, 9, 11	DTACKN low after D0 – D15 valid data	625		ns
11	7	CSN high after ASN, UDSN, LDSN, A1 and A2		10	ns
12	7,8	D0 – D15 hold after CSN high	10		ns
13	7,8	Local high after CSN high	10		ns
14	7	DTACKN high after ESN high	10	120	ns
15	7, 8	CSN low time	80		ns
16	7, 8	A1 and A2 HOLD after CSN high	0		ns
17	8	DDIR low after CSN low		1	µs
18	8	D0 – D15 valid before DTACKN low	300		ns
19	8	R/WN low before CSN high	0		ns
20	8	R/WN low after CSN high	0		ns
21	9	IACKN low after last of ASN AND LDSN		30	ns
22	9	IRQN high after IACKN low		140	ns
23	9	Local low after IACKN low		1	µs
24	9	D0 – D15 valid after last low of ASN, LDSN, IACKN	1		µs
25	9	IACKN low time	1.5		µs
26	9	D0 – D7 hold after LDSN high	20		ns
27	10	BRN high after BGACKN low		1.2	µs
28	11	OWNX low after BGACKN low		1.2	µs
29	11,12	D0 – D15 valid before either VAS or LAS low	65		ns
30	11	DDIR low after OWNX low		340	ns
31	11,12	D0 – D15 valid after either VAS or LAS low	20		ns
32	11,12	LAS low before ASN low	0		ns
33	11	ASN, data strobes width low (read)/ASN write	160		ns
34	11,12	Local low after data strobes low		20	ns
35	11,12	D0 – D15 valid after data strobes high	30		ns
36	11	Local high after data strobes high	30		ns
37	11	DDIR high after ASN high	20		ns
38	11,12	OWNX high after ASN high		400	ns
39	11,12	BGACKN high after OWNX high		30	ns

Preliminary

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTICS	TENTATIVE LIMITS		UNIT
			Min	Max	
40	12	Data strobes with low	80		ns
41	12	R/WN low before ASN low	35		ns
42	12	D0 – D15 valid before DTACKN low	50		ns
43	12	R/WN low after data strobes high	30		ns
44	13	D0 – D15 valid before EN0 high	40		ns
45	13	D0 – D15 valid after EN0 low	40		ns
46	14	D0 – D15 valid after EN1 low		40	ns
47	14	D0 – D15 valid after EN1 high	0	50	ns
48	15	WRGATE high after SECTOR/INDEX high	TBD		
49	15	WRDAT or WRCIK valid after WRGATE high	TBD		
50	16	REDAT high before WCLOCK/RCLOCK high	20		ns
51	16	REDAT high after WCLOCK/RCLOCK high	20		ns
52	18	RERUNN low before DTACKN high	20		ns
53	18	RERUNN low after DTACKN high	320		ns

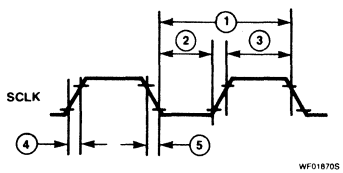


Figure 5. Input Clock Timing

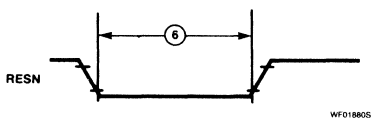


Figure 6. Reset Timing

Preliminary

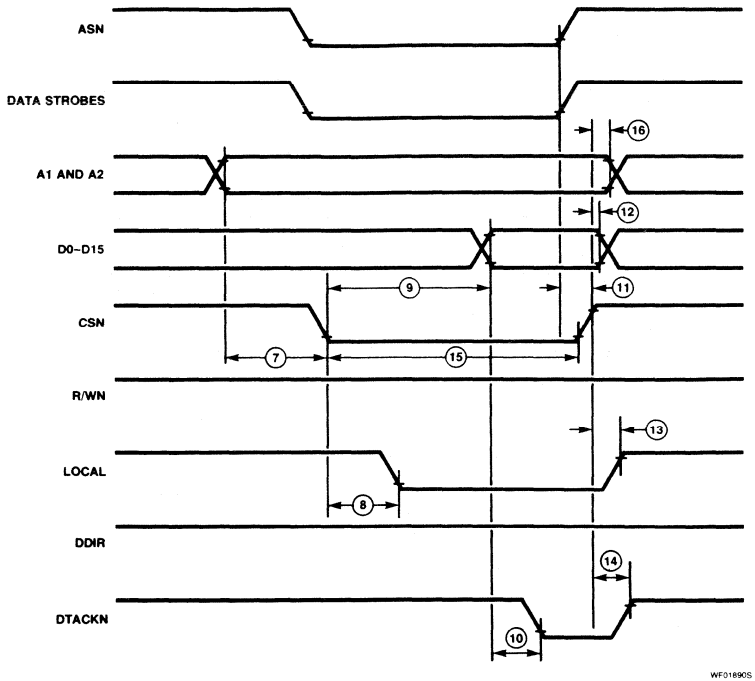


Figure 7. IMDC Register Read Cycle

Preliminary

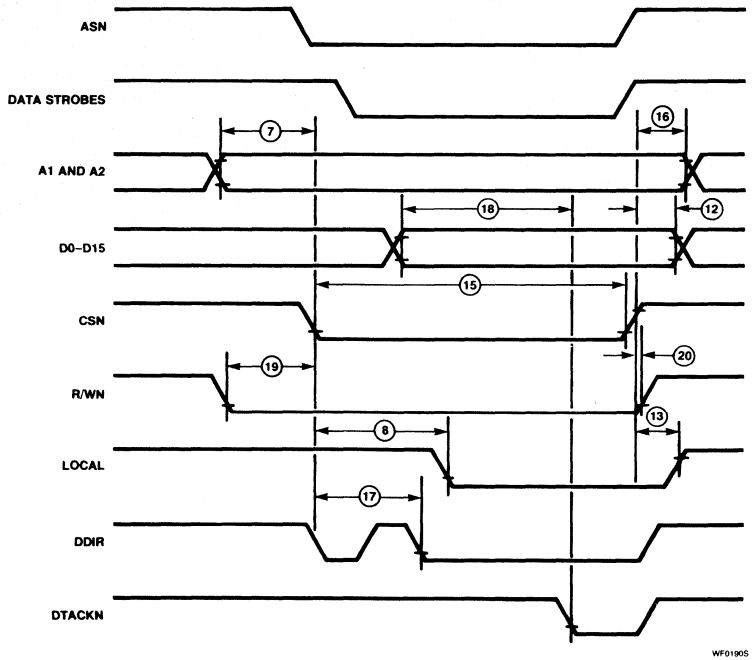


Figure 8. IMDC Register Write Cycle

Preliminary

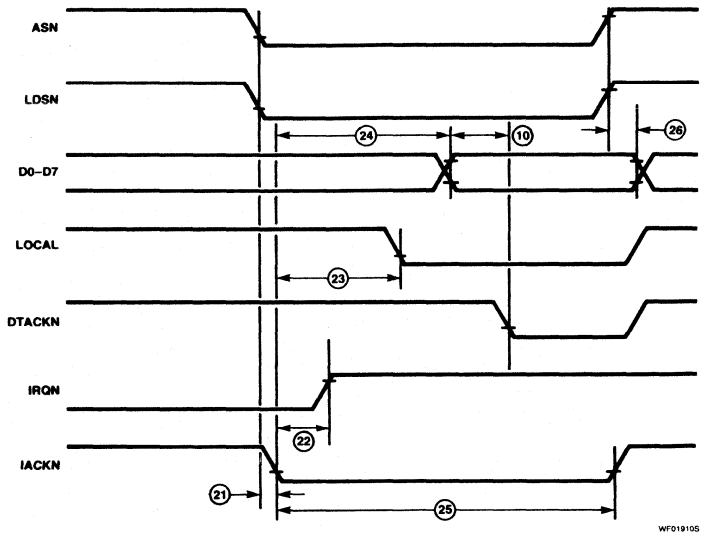


Figure 9. Interrupt Acknowledge Cycle

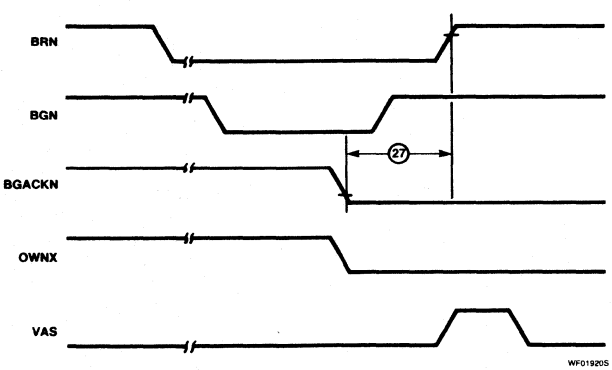


Figure 10. Bus Arbitration

Preliminary

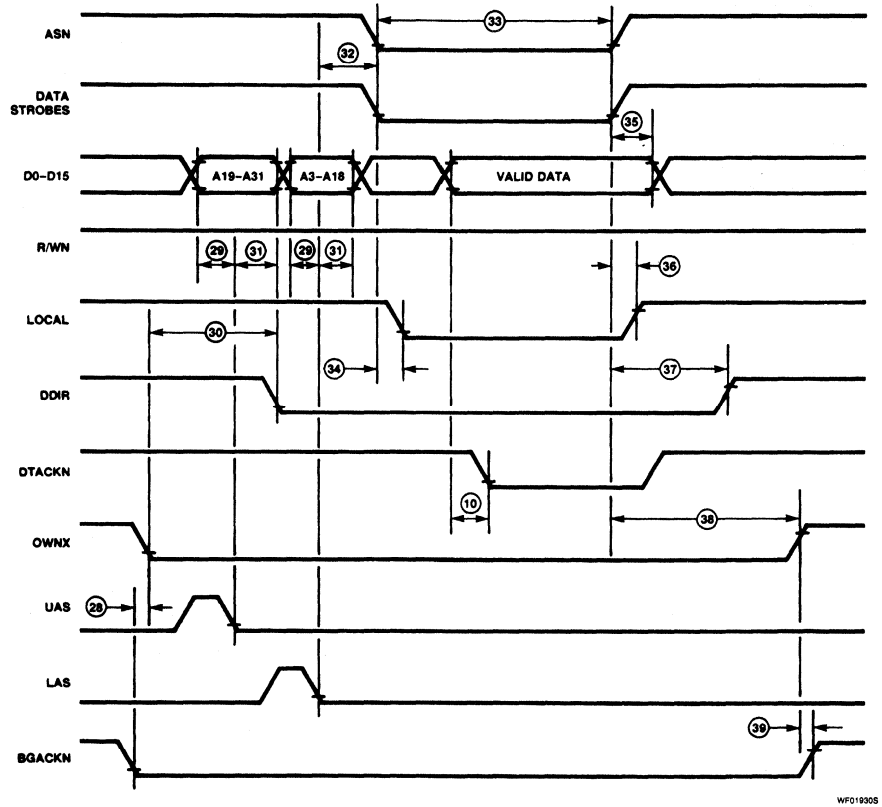


Figure 11. Single 16-Bit DMA Read From Memory Cycle

Preliminary

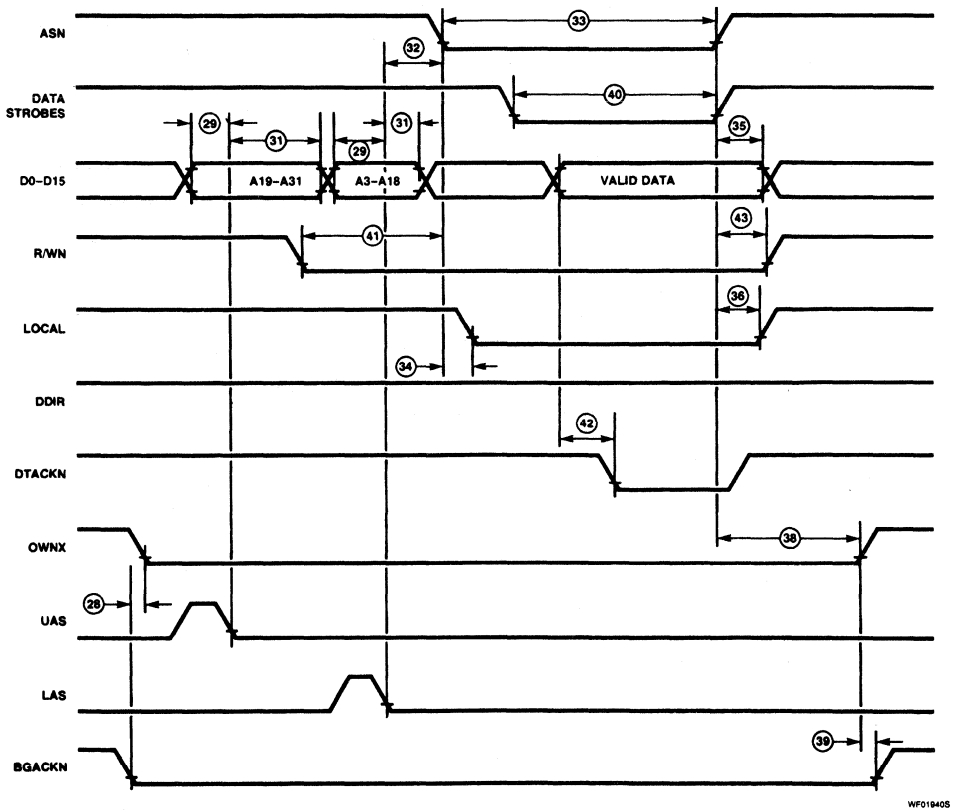


Figure 12. Single 16-Bit DMA Write to Memory Cycle

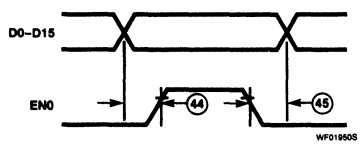


Figure 13. IMDC Output Port Timing

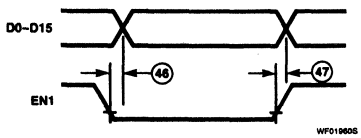


Figure 14. IMDC Input Port Timing

Preliminary

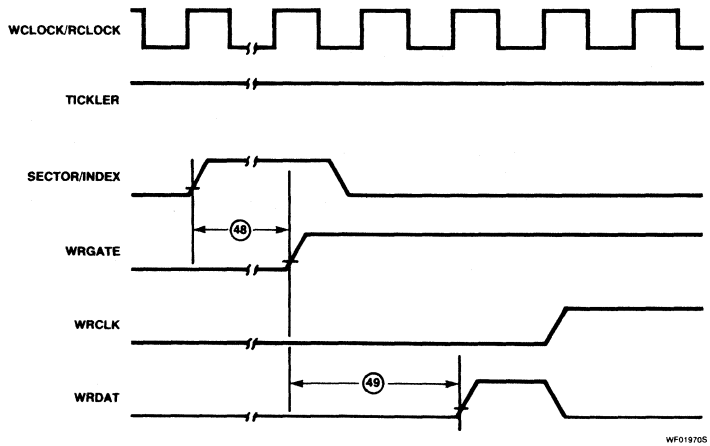


Figure 15. Write Operation

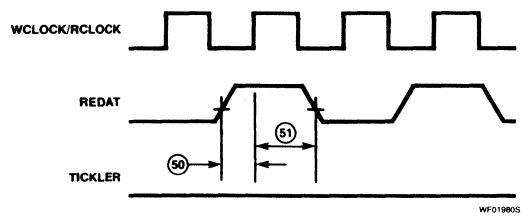


Figure 16. Read Operation

Preliminary

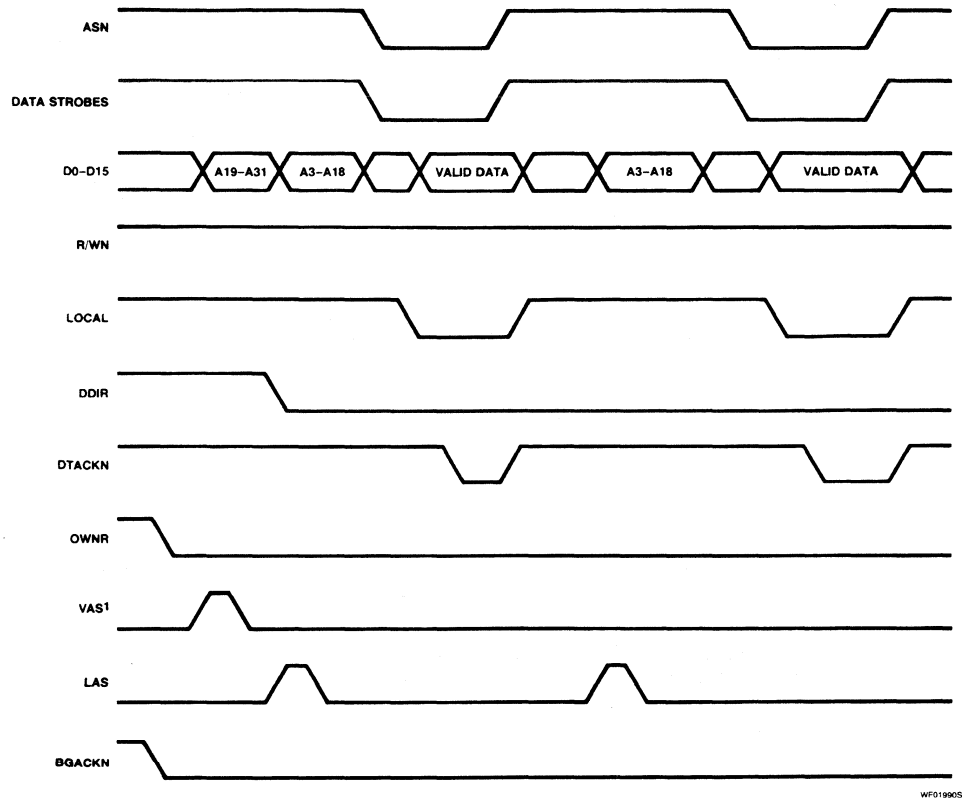


Figure 17. Example of Multiple DMA Read Cycles

[1] If the high order address does not change, VAS will not be asserted.

Preliminary

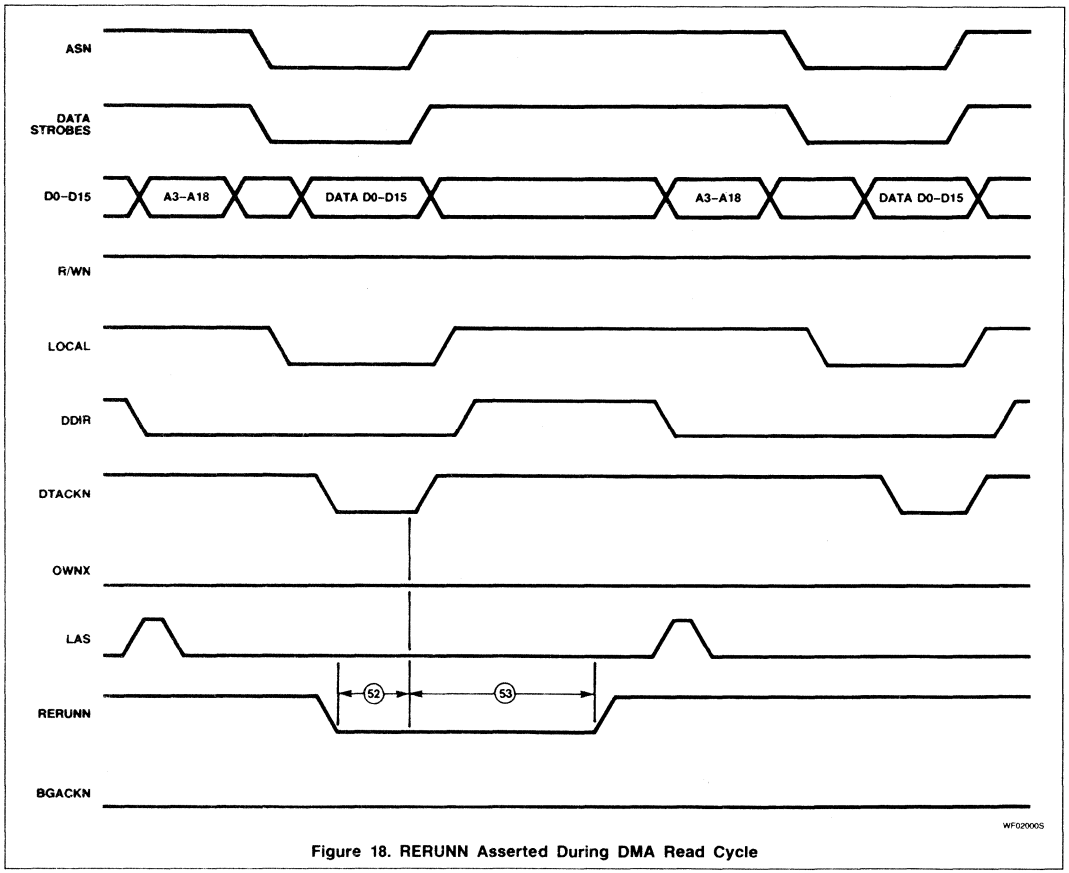


Figure 18. RERUNN Asserted During DMA Read Cycle

DISK PHASE LOCKED LOOP (DPLL)

Originally published by Signetics November 1983

Advance Information

DESCRIPTION

The SCB68459 Disk Phase Locked Loop (DPLL) is a 20 pin bipolar chip which complements the SCN68454 Intelligent Multiple Disk Controller (IMDC). Together, with an external Voltage Controlled Oscillator (VCO), they provide all the functions to control up to four disks with SA800, ST500, or SA1000 type interfaces.

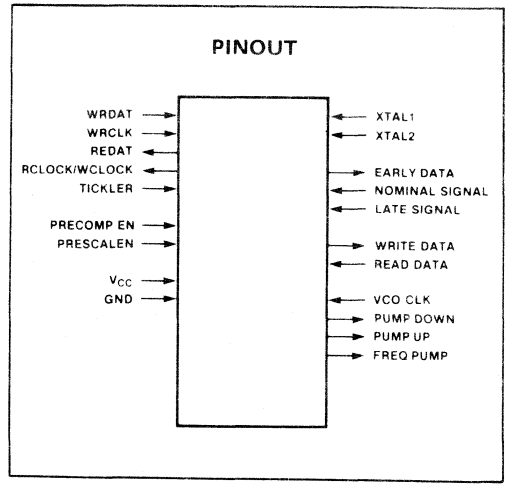
The DPLL uses an external VCO for the variable clock rate which tracks the read data from the disk unit. This VCO can be any device which properly interfaces to the DPLL.

The IMDC can control up to four disks; if these are all the same only one DPLL is needed. However, if different disk types are driven by the same IMDC, then a DPLL is needed for each disk drive type.

The SCB68459 DPLL operates by producing an ocillator frequency to match the frequency of an input signal. In this locked condition, any slight change in the input frequency (called jitter) will appear as a change in phase between the input frequency and the VCO frequency. This phase shift then acts as an error signal to change the frequency of the local DPLL VCO to match the input frequency.

The SCB68459 is constructed using Signetics extended performance logic (EPL) bipolar technology and is contained in a 20-pin dual-in-line-package.

PINOUT



FEATURES

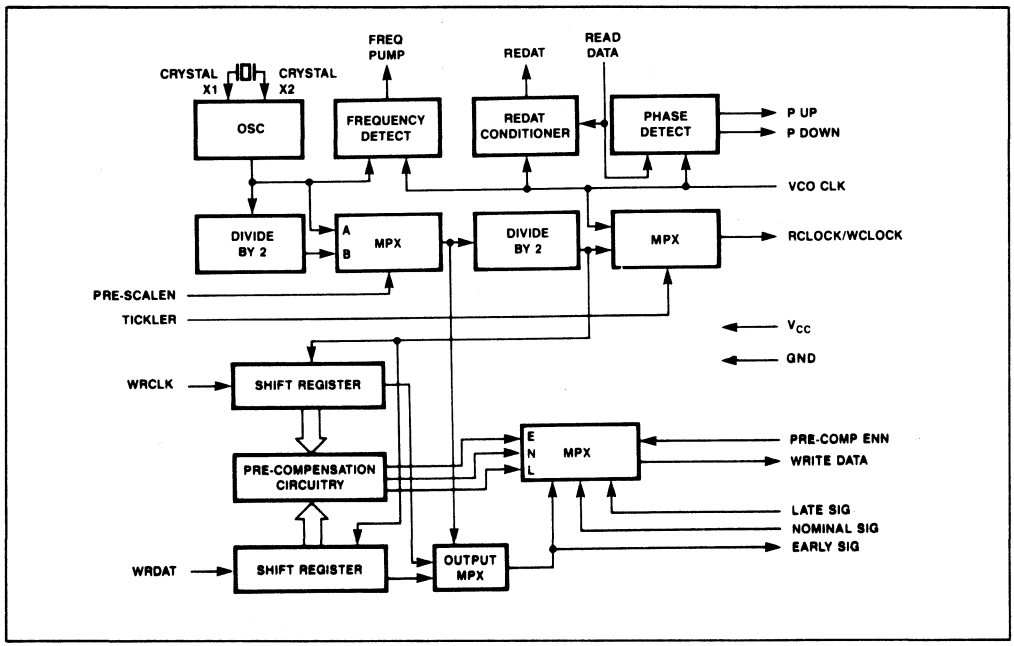
- Supports composite data rate of 100KHz to 10MHz
- On chip multiplexer for read and write clock
- Supports MFM and FM data formats
- Generates synchronous clock for read data
- Built in pre-compensation circuit
- Phase detector and frequency detection for improved operation
- External loop gain control
- Minimal amount of external components required for operation
- Single +5 volt power supply
- Crystal controlled write clock

ORDERING CODE

Packages	V_{CC} = 5V ±5%, T_A = 0 to 70°C
Ceramic DIP	SCN68459CAI20
Plastic DIP	SCN68459CAN20

Advance Information

BLOCK DIAGRAM



PIN DESCRIPTION

The pin designation table describes the function of each of the pins of the DPLL. Signal names ending in "N" are active low. All other signals are active high.

PIN DESIGNATION

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
WRDAT		I	Write Data Pattern. Active high. WRDAT is a non-return to zero (NRZ) input which contains the data information; i.e., it is the polarity of the WRITE DATA to be output in the second half of the bit cell time.
WRCLK		I	Write Clock Pattern. Active high. WRCLCK is an NRZ input which contains the clock information; i.e., it is the polarity of the WRITE DATA to be output during the first half of the bit cell time.
PRE-SCALEN		I	Pre-Scale. Active low. When PRE-SCALEN is low, the crystal frequency is divided by 4 (the normal divide by two plus an additional divide by 2) before being used as WRCLOCK. When PRE-SCALEN is high, the crystal frequency is not pre-scaled. Typically used to encode/decode both FM and MFM with the same crystal frequency.

Advance Information

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
TICKLER		I	Tickler. Active high. TICKLER controls the operation of the RCLOCK/WCLOCK output signal. If TICKLER is high, internal logic routes the crystal controlled clock to the output. If TICKLER is low, the PLL synchronized to the incoming READ DATA will generate RCLOCK. This clock is then routed to the output signal RCLOCK/WCLOCK.
RCLOCK/WCLOCK		O	Write Clock/Read Clock. RCLOCK is the output when the TICKLER signal is low. As an output, it is the clock phase locked onto the incoming READ DATA signal. RCLOCK is twice the data frequency. WCLOCK is the output of the crystal oscillator, divided internally by 2, and will output continously while the DPLL is enabled and the TICKLER signal is high. WCLOCK defines the bit cell frequency.
PRE-COMP ENN		I	Pre-Compensation Enable. Active low. When PRE-COMP ENN is high, no write compensation is applied to the outgoing data. When it is low, the data stream is write pre-compensated.
FREQ PUMP		O	Frequency Pump Error Signal. Current output which is summed through a resistor along with P UP and P DOWN. The three outputs are then used to charge or discharge a capacitor which generates an error voltage for the VCO.
P DOWN		O	Pump Down Error Signal. Current output which is summed through a resistor along with FREQ PUMP and P UP. The three outputs are then used to charge or discharge a capacitor which generates an error voltage for the VCO.
P UP		O	Pump Up Error Signal. Current output which is summed through a resistor along with FREQ PUMP and P DOWN. The three outputs are then used to charge or discharge a capacitor which generates an error voltage for the VCO.
VCO CLK		I	Variable clock which is generated by an external VCO.
CRYSTAL X1		I	Crystal 1. Input connection provided to attach a crystal for the internal oscillator. The crystal frequency is defined to be twice the data bit rate.
CRYSTAL X2		I	Crystal 2. Input connection provided to attach a crystal for the internal oscillator. The crystal frequency is defined to be twice the data bit rate.
EARLY SIG		O	Early Data Signal. Active high. EARLY SIG is used as the input to a user supplied delay line or delay circuit. The delays are determined by the requirements of the particular disk unit to which the DPLL is attached. If pre-compensation delays are enabled by the signal PRE-COMP ENN, this signal together with NOMINAL SIG and LATE SIG are used to generate the WRITE DATA signal.

Advance Information

MNEMONIC	PIN NO.	TYPE	DESCRIPTION
NOMINAL SIG		I	Nominal Data Signal. Active high. This input is used by the DPLL together with the EARLY SIG output and the LATE SIG to generate the write pre-compensation delays. The user should insert a delay line between the EARLY SIG output and the NOM SIG input and between the output of the first delay line and the LATE SIG input (these delays being the delay between an early and nominal signal or a nominal and late signal, respectively). The values of these depend on the disk drive to be used. If pre-compensation is not enabled, NOMINAL SIG is used to generate the WRITE DATA signal. In systems where pre-compensation is never used, the EARLY SIG output should be connected to the NOMINAL SIG input.
LATE SIG		I	Late Signal. Active high. This input is used by the DPLL together with the NOMINAL SIG to generate the write pre-compensation delays. The user should insert a delay line between the NOM SIG input and the LATE SIG input. The value depends on the disk drive to be used.
READ DATA		I	Read Data. Active high. This is the composite clock and data read from the disk unit. It is used as the incoming signal for the phase lock loop.
WRITE DATA		O	Write Data. Active high. This is the output signal, write pre-compensated if so enabled, formed from the WRDAT and WRCLK inputs.
REDAT		O	Composite Read Data. Active High. REDAT is the output signal which reflects the READ DATA input signal but synchronized to the RCLOCK generated by the internal PLL. The REDAT signal will be asserted for one period of the RCLOCK signal following the assertion of READ DATA.
V_{CC}		I	+5 volt $\pm$5% power input.
GND		I	Signal and power ground input.

DPLL FUNCTIONS

Read/Write Clock

The DPLL provides a read/write clock for the IMDC. This clock is either derived from a crystal or from a PLL locking onto the combined data and clock stream from the disk.

Generation of the Output Data

The DPLL inputs two pieces of information, the NRZ data and clock information bits in parallel, from the IMDC. It combines them into the same bit cell by issuing the clock during the first half of the first half of the bit cell time, and the data information during the first half of the second half of the bit cell time. This information then becomes the WRITE DATA signal which goes to the disk drive.

Pre-Compensation

In addition, the DPLL provides facilities for pre-compensating the data during the writing of the data to disk. The write pre-compensating algorithm is built into the chip. Note that this feature can be disabled where necessary.

The logic in the DPLL does the pre-compensation algorithm in a slightly different manner than with a ROM lookup table. If the different data and clock patterns which require compensation are drawn out, only two patterns are found. The two patterns are a series of ones followed by a zero, or a series of zeros followed by a one. By using a shift register and simple logic, the signals necessary for gating early, late, and nominal can be determined (the last being the absence of the previous two). This is the method which is implemented in the DPLL.

Advance Information

PLL Function

With a Phase Locked Loop (PLL) circuit in the read chain, the data can be recovered more accurately. The PLL provides a clock signal which is derived from the read data stream and tracks it through a reasonable variation. The clock signal is used by the decoding hardware to sample the read data stream.

Delay Lines

The delay lines referred to in this document can be purchased as a single unit or constructed with logic and passive components. The delay line, typically, has a single input with several outputs. The outputs reflect the input but delayed by some period of time. A delay line for an eight inch MFM floppy disk drive, for example, would have each output delayed 175 nanoseconds from the other. The first output is delayed 175 nanoseconds from the input and the second output is delayed 175 nanoseconds from the first output. The delay time actually used in each application would be the function of the disk drive and specified by the manufacturer.

OPERATION

The PLL in the DPLL is intended to be used with an external VCO. As shown in figure 1, the DPLL interfaces between the IMDC and the disk unit. The only other attachments are the resistors and capacitor for the external VCO. These values should be chosen for the operating characteristic required by the system.

For IBM System 34 interfaces, the decode of side zero, track zero, would be used for the PRE-SCALEN input. This input would then automatically divide the crystal controlled clock output for RCLOCK/WCLOCK.

The crystal used with the DPLL should be twice the data rate for the interfaced disk unit.

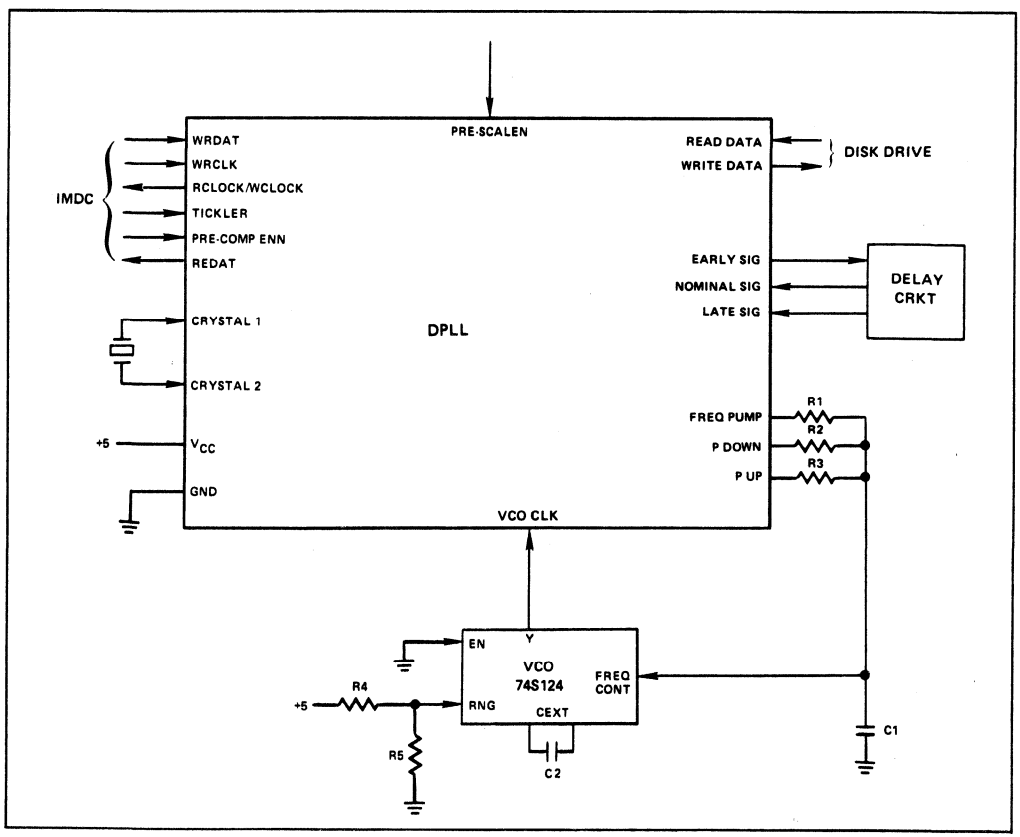
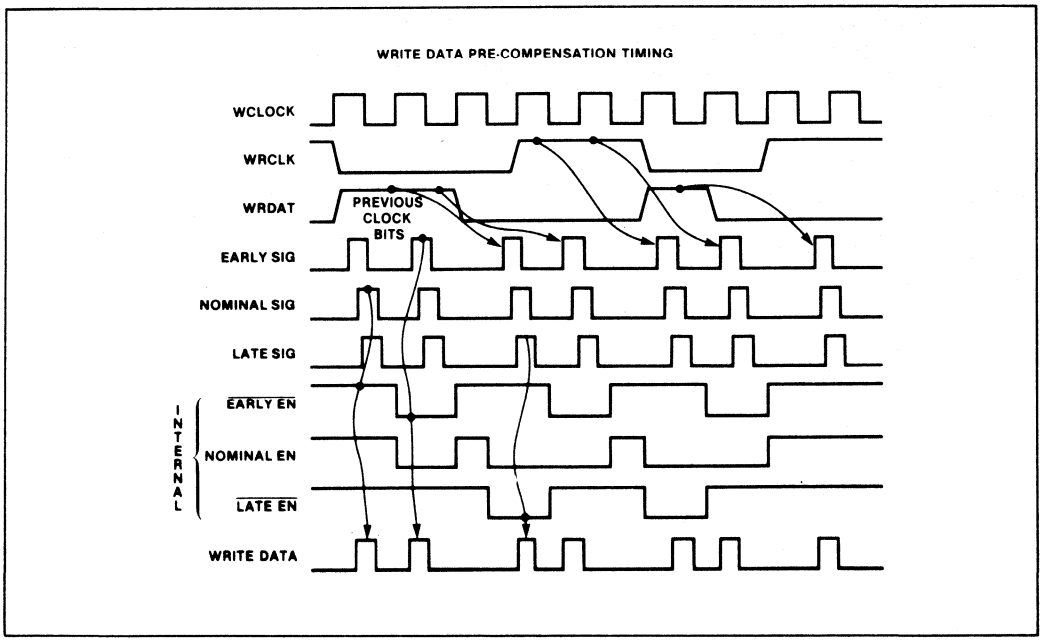
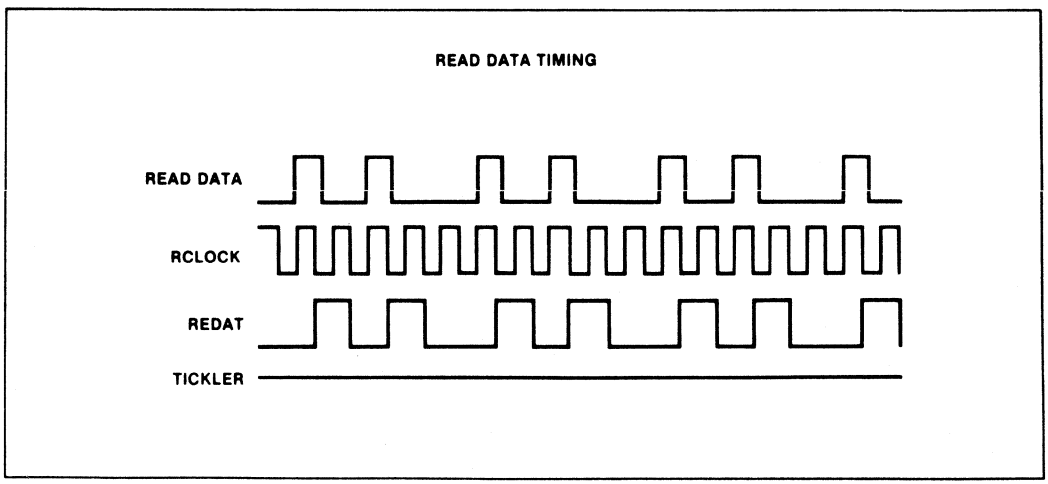


Figure 1. Typical Application

Advance Information





Disk controller supports both rigid and floppy drives

Two ICs let you build a disk controller that supports as many as four rigid- and/or floppy-disk drives, and you can select from a number of different disk and drive-interface formats.

Guy Thomsen, *Signetics Corp*

Many designers are developing systems that use hard disks for on-line mass storage and some other medium to back up the disks. Because it's compact, portable and inexpensive, the floppy disk has been a popular back-up choice. But until now, using the floppy disk has meant that you had to develop two separate controllers—one for each drive type.

Using a new 2-chip set, however, you can design a single controller to service both rigid and floppy disks. The two ICs are the SCN68454, an intelligent multiple-disk controller, and the SCB68459, a phase-locked loop for the disk drives. You can use the pair of chips to control as many as four rigid or floppy disks in any combination. The ICs can work with any drive that has a Shugart SA850, SA1000 or Seagate ST506 interface. (For more details on the controller and phase-locked loop, see the **box,** "Take a closer look at the chip set.")

If you want to employ these chips, though, the controller will affect your overall system design, especially its memory usage and the software drivers that control data transfers. Adding the hardware to your system, on the other hand, is an easy process, and once you understand the hardware, the controller's operation and its effect on your software becomes clear. So it's best to begin with a hardware implementation that can serve as a model for understanding the controller's unique way of operating.

Simple controller hardware

Using the IMDC and DPLL hardware in the design of a controller is rather straightforward. As **Fig 1** shows, you need only a few ICs—some octal buffers, an address decoder, a multiplexer, and a couple of delay lines and voltage-controlled oscillators (VCOs)—in addition to the IMDC and two DPLLs. Using conventional ICs, you'd need more than 90 devices to provide the same features found in the IMDC/DPLL-implemented design.

The controller's interface to the 68000-based host system's bus consists of five octal buffers: three 74LS373s for address and function codes and two 74LS245s for data. Address lines A_1 and A_2 and all other control lines can be tied directly to the system bus. Once the host has activated the IMDC by sending the IMDC's address to the address decoder (which activates the IMDC's chip-select (CS) line), lines A_1 and A_2 directly address the IMDC's internal registers.

Tied to the controller side of these five buffers is a 16-bit bus, which is called the local bus and is controlled by the IMDC. When the IMDC wants to send an address to the host, it first puts the high-order addresses A_{19} through A_{23} and function codes FC_0 through FC_2 on bus lines D_8 thorugh D_{15} and asserts the upper address strobe (UAS). IC_5 retains the data after UAS returns Low.

Next, the IMDC puts the lower-order address codes A_3 through A_{18} on the bus and asserts the lower address strobe (LAS). When LAS is driven Low, IC_3 and IC_4 retain the address. Finally, the IMDC puts the remaining address bits on lines A_1 and A_2 and sends the address through a normal 68000-bus cycle using the address strobe, $\overline{AS}$, and the data strobes, $\overline{UDS}$ and $\overline{LDS}$. If, during a consecutive addressing operation, the higher-order address lines don't change, the IMDC doesn't reload the upper address lines.

The local bus transfers data, as well, to and from the

Reprinted from **EDN** October 4, 1984

host system. When the IMDC uses DMA to read data from host memory, for example, it first latches that data into IC_1 and IC_2; the state of the data-direction (DDIR) line determines which port of these bidirectional buffers will accept the data for buffering, and the LOCAL line enables the buffers. Then, using the local bus, the IMDC moves the data to temporary storage in the IMDC's 128k-byte first-in, first-out (FIFO) buffer before transferring the data to a disk. A write operation is just the opposite: data from the disk is sent to the FIFO, then to the bidirectional buffers, and then to the host memory.

The local bus also carries control information between the IMDC and the disk units. EN_0's going High latches the data into IC_6 and IC_7. With $\overline{EN_1}$ Low, buffer IC_8 passes information back to the IMDC.

Data is written to or read from a drive by means of the DPLL. In **Fig 1**, two DPLLs are used: one for a slow read/write data path (floppy-disk drive), and the other for a fast read/write data path (rigid-disk drive). The example uses two parts because different drives can require different read-data rates and associated voltage-controlled-oscillator (VCO) timing, and because the two write clocks require different crystals. If the

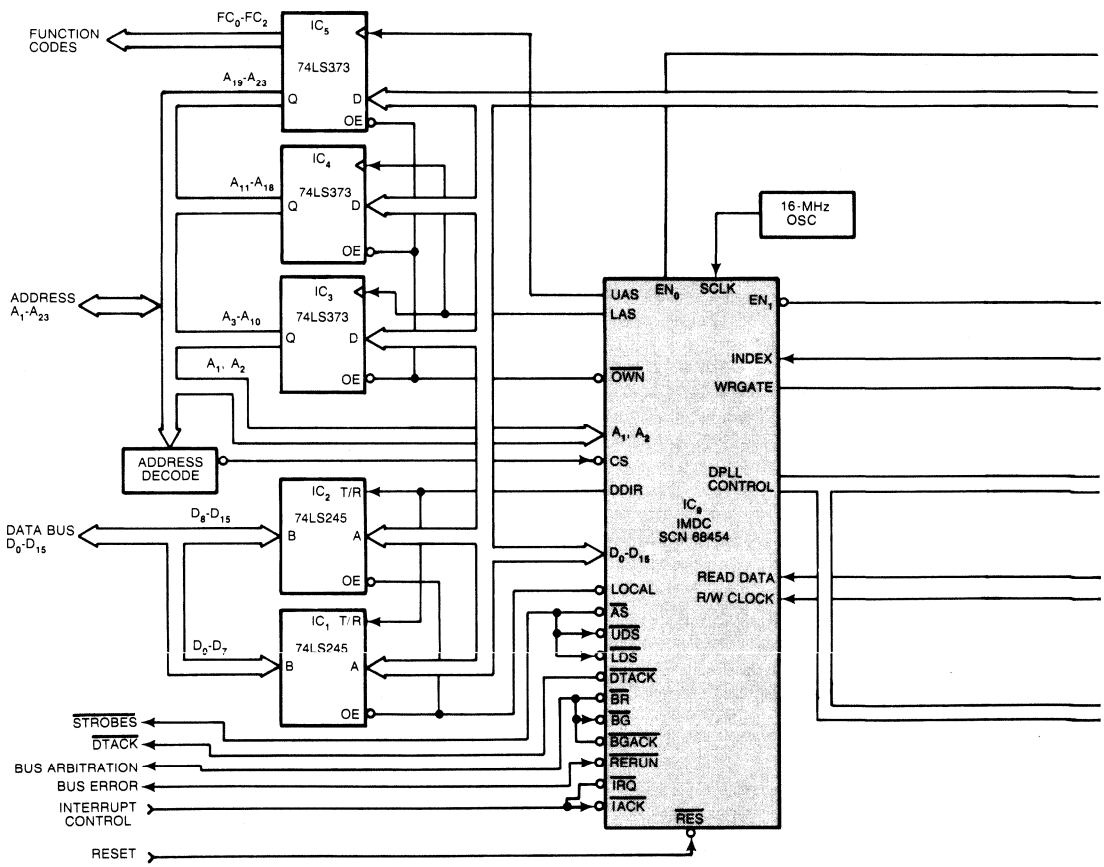
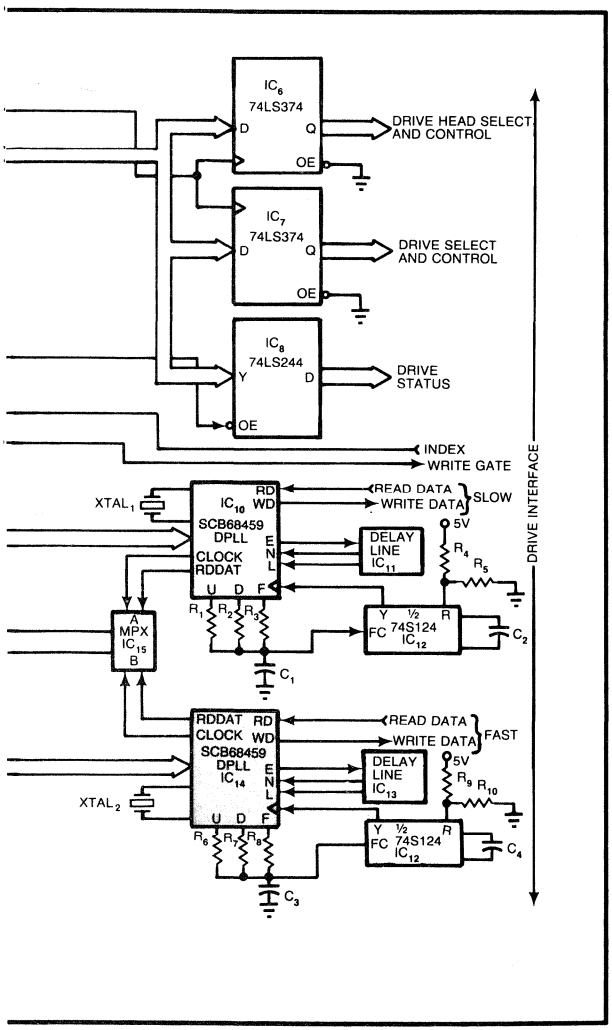


Fig 1—This 2-drive controller *uses few ICs because its controller chip set is made up of an Intelligent Multiple Disk Controller (IMDC) and two Disk Phase-Locked Loop (DPLL) ICs.*

data rates for both types of drives attached to the IMDC were the same, then the circuit would only require one DPLL. In any case, you only require one IMDC to control four disk units.

In this case, each DPLL uses a separate VCO, ½ of a 74S124. You attach the VCO to the DPLL via three resistors and a capacitor. The resistors attach to the pump-up, pump-down and frequency-pump pins of the DPLL. The voltages that the DPLL pumps across these resistors are summed by the capacitor. The sum point of the four passive devices attaches to the VCO's frequency control pin.



In addition to the four passive devices used for frequency control, passive elements control other VCO parameters. Two resistors (you could use one trim pot instead) set the range control, and a capacitor sets the range's center frequency.

The final part that the DPLL requires is the crystal. This crystal must have a frequency that is twice the data rate of the disk drive or drives attached to the DPLL. The DPLL function requires only three ICs and eight passive components. Many other designs would require amplifiers and a handful of components to accomplish the same function.

You need the multiplexer, IC_{15}, when you incorporate two or more DPLLs into a design. IC_{15} selects the Read Data and Read/Write Clock from the DPLL for the drive chosen. The other control signals can go directly to each DPLL.

Initialize first

Before the IMDC performs any operation, you have to initialize three areas: one in the controller and two in the host system. These areas are the IMDC's internal registers and two special areas of the host memory—the event control area (ECA) blocks and the ECA pointer table. These three areas are critical to the controller's unique way of performing: By writing to the IMDC's registers, the host system tells the controller what operations to perform and, by looking at the ECA pointer table and the ECA blocks, the IMDC knows how to perform them.

There are seven addressable registers in the IMDC (**Fig 2**). To initialize them, the host must load the first four registers—the ECA registers—with the address of the ECA pointer table. It then loads the fifth register—the Interrupt Vector register—with the vector-priority number assigned to the IMDC. The last two registers are initialized just prior to actual operation; before initializing them, you must initialize the ECA pointer table and the ECAs themselves.

The ECA pointer table contains the addresses of the ECA blocks, of which there can be as many as four—one for each of four drives. The block addresses are arranged by drive number in ascending order. The ECA block addresses need not be contiguous; the only requirement is that the IMDC be able to address each.

The ECA blocks must also be initialized. That is, each block must contain the drive parameters that the IMDC needs in order to determine whether the drive is rigid or floppy. Normally, once you set up these areas, you shouldn't have to change them again.

While there is normally a separate ECA block for each drive, you can use an ECA block for more than one disk unit if you take proper care. If there are common parameters in ECA blocks, save those parameters

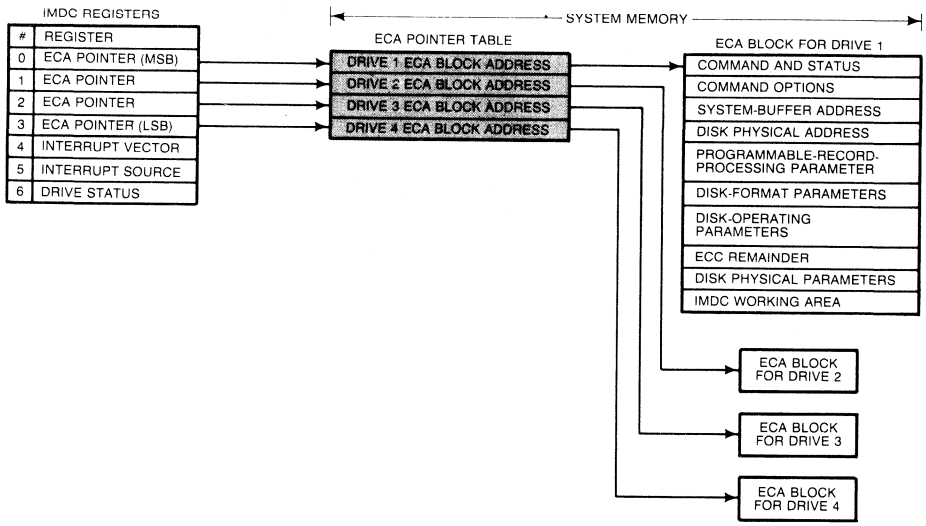


Fig 2—Event Control Area (ECA) addressing *is the responsibility of both the Intelligent Multiple Disk Controller (IMDC) registers and system memory. The first four registers are pointers to the ECA pointer table, which contains the ECA block addresses.*

separately from the unique parameters that are associated with a particular disk's physical address. After you access the unique parameters, branch to the common parameters.

Within the ECA block, eleven IMDC commands provide read and write data transfers (**table 1**). Status gets updated at the completion of each requested operation. The status consists of two areas: a main status and an extended status. For each command completion, it is necessary to check only the main status to determine whether an error is posted. If there is no error, the extended status provides additional information about the operation.

The System Buffer Address and Disk Physical Address define where to transfer data from and to during a read or write. The remainder of the ECA block defines special parameters for each command and drive type. The IMDC uses these bytes to determine whether the disk unit is rigid or floppy, and the extent of storage capabilities. Included are the parameters that define step rate, head load time (if required) and other physical features.

In the IMDC, there are only two registers that the system software uses during disk operations: the Interrupt Source and Drive Status registers. To request an IMDC operation, the host software must set one of four bits in the Drive Status register. Each bit corresponds

TABLE 1—IMDC COMMANDS

COMMAND MNEMONICS	HEX CODE	DESCRIPTION
WRMS	00	WRITE MULTIPLE SECTOR
WRDD	01	WRITE WITH DELETED DATA FLAG
VER	10	VERIFY SECTOR DATA
REMS	11	READ MULTIPLE SECTOR
PRP	12	PROGRAMMABLE RECORD PROCESSING
TSR	13	TRANSPARENT SECTOR READ
RETD	20	READ IDENTIFIER
FORM	40	FORMAT TRACK
CALB	41	RECALIBRATE TO TRACK ZERO
CORR	81	CORRECT DATA WITH ECC REMAINDER
DIAG	80	DIAGNOSTIC

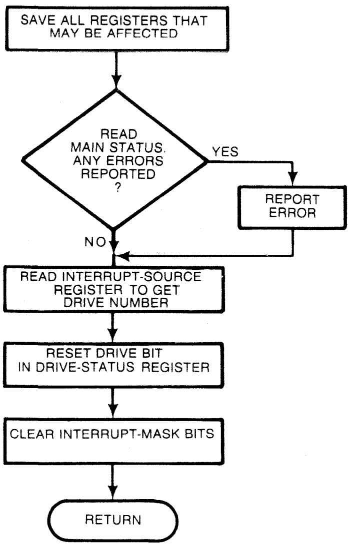


Fig 3—This flowchart contains *all the interrupt handling for a simple disk-controller system.*

to one of the four drives. Setting a bit flags the IMDC to use the ECA pointer register plus an appropriate offset to get the ECA block address. The IMDC then uses this address to obtain the what, where and how for the current request.

When the IMDC completes the posted command, it sets the appropriate bit in the Interrupt Status register and asserts the interrupt line to the host processor. The system software then reads the ECA status to determine proper command completion; the cycle continues as necessary. You can set each of the bits in the Drive Status register independently of the others. Resetting a bit before an operation is complete aborts the request and sets an interrupt. **Fig 3** shows a sample flowchart. **Fig 4** contains an interrupt service routine for the IMDC written in 68000 μP assembly language.

Format different drive types

The IMDC lets you format disks to several different standards. The device supports both the IBM single-density (IBM 3740) and double-density (IBM System 34) formats, and it can format both hard- and soft-sectored disks. The user can also specify either a 1- or 2-byte cylinder field within the sector header. This accommodates the next generation of rigid disk units which may have more than 256 cylinders.

Each of the format options has a fixed number of bytes that are repeated within each sector; the IMDC automatically writes those bytes. The IMDC fetches from a table in memory the bytes that vary, such as the cylinder number, head and sector number. The System Buffer Address in the ECA table points to this table.

For each sector on each track of a floppy disk, the software driver first puts into ECA memory the cylinder number, side, sector number amd record length. Using DMAs, the IMDC then puts the information into the FIFOs during the format operation. Because the user places the information into the ECA, he or she can specify the sector mapping on the track. This can take place in any order the user wants, and it links together as many consecutive sectors as the system software requires. The IMDC can read consecutively numbered sectors over the total track length.

The information for a rigid-disk format is the same as for the floppy-disk format, except that there is no sector length placed in the sector header. With a rigid disk, formatting takes place one track at a time. This allows the user to perform other disk operations without waiting for the whole disk to be formatted. For some drives, the formatting time is several minutes.

Learn to handle bad sectors

The IMDC provides the user with a bad-sector feature. If, after formatting a track, tests show that an area has an unreliable sector, the IMDC provides a bad-sector flag, along with an alternative physical address. During a read or write to that sector, the IMDC finds the flag, reads the alternate sector address and

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                 INTERRUPT SERVICE ROUTINE FOR IMDC                     ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
IMDC-INTR  MOVE.L   DO,-(SP)          ;SAVE REGISTER
           MOVE.B   MAIN-STATUS,DO    ;GET MAIN-STATUS BITS
           BEQ.S    INTR-CONT         ;SKIP ERROR REPORT IF OK
;
           BSR.L    ERROR-REPORT      ;GO TO ERROR REPORTING ROUTINE
;
INTR-CONT  MOVE.B   INTR-SOURCE,DO    ;GET THE FINISHED DRIVE NUMBER
           EOR.B    DO,DRIVE-STATUS   ;CLEAR DRIVE-BUSY BIT
;
           MOVE     SR,DO             ;GET 68000 STATUS BITS
           AND.W    #INTR-MASK,DO     ;CLEAR INTERRUPT-MASK BITS
           MOVE     DO,SR             ;RETURN STATUS BITS TO 68000
;
           MOVE.L   (SP)+,DO          ;RETURN SAVED REGISTER
;
           RTR                        ;RETURN FROM INTERRUPT
```

Fig 4—This interrupt service routine *for the Intelligent Multiple Disk Controller is written in 68000 assembly-language code.*

goes to that sector. The IMDC then moves the drive head to that sector, performs a seek operation (if one is required by a read operation), and completes the transfer operation in the normal way at that location. Then the IMDC returns to the track with the bad sector and continues the read operation at the next sector.

The IMDC's ability to select alternate sectors is especially useful during multiple-sector data transfers.

Take a closer look at the chip set

The SCN68454 Intelligent Multiple-Disk Controller (IMDC) illustrated in **Fig A** is a microprogrammed disk-control subsystem that employs a memory-to-memory architecture. Bus compatible with the 68000 μP family and its vectored interrupt structure, the IMDC has 31-bit address lines for the DMA transfer, a 128-byte first-in, first-out (FIFO) buffer, a 31-bit address counter and user-specified data transfers of either 8 or 16 bits to the host data bus. Housed in a standard 48-pin DIP, the IMDC needs only a single 5V power supply.

The IMDC is the interface to both the host system and the disk units. The IMDC and host system use the IMDC's seven internal registers for communication about the task to be performed. After the IMDC accepts a command from the host, an on-chip direct-memory-access (DMA) controller takes charge of all transfers between system memory and IMDC. The IMDC uses its on-chip FIFO buffer to provide temporary storage during data transfers.

The IMDC's disk-interface section contains the read and write logic, the serial/deserial register, the error check and correction (ECC) register, and the disk input/output (I/O) control logic. The **table** shows the input and output lines.

Complementing the IMDC is the SCB68459 Disk Phase-Locked Loop (DPLL), a 20-pin bipolar IC. Like the IMDC, it requires only a single 5V power supply. The block diagram **(Fig B)** shows the DPLL's internal architecture.

The DPLL uses an external voltage-controlled oscillator to provide a variable clock rate to track the read data from the disk unit. The on-board crystal oscillator also generates the write clock. The DPLL operates with a composite data rate of 100 kHz to 10 MHz for precision management of the bit stream flowing between the disk drive(s) and the host system. Precompensation logic is built into the chip; designers can use this precompensation to write-compensate the modified-frequency-modulation (MFM) data to the disk unit.

However, the devices are not limited in scope to use with the 68000 μP family. Designers can integrate the IMDC and DPLL for use with any μP architecture that employs either an 8- or 16-bit data

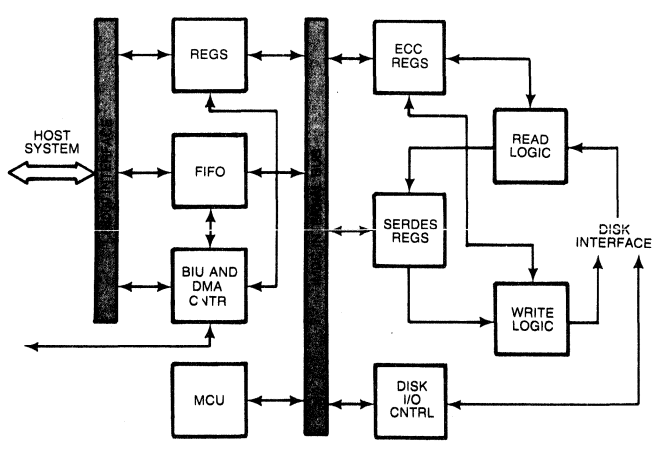
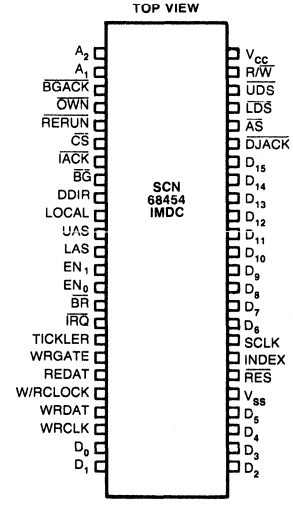


Fig A—The Intelligent Multiple Disk Controller *provides an interface between the host system and the disk units.*



PIN CONFIGURATION FOR IMDC

The user is not aware of any interruptions in the actual data transfer. It also relieves the system software of the responsibility of maintaining an alternate-sector map on the disk and performing the extra operations required during the data transfers.

An unusual feature built into the disk controller is that it can search for a character string and then perform an operation. With this feature (the Program-

bus. In any system design, expect a relatively low parts count. The 2-chip set integrates the equivalent functions of 90 or more ICs. Because of this high level of integration, the chips reduce pc-board space by as much as 90%. Correspondingly, manufacturing costs are reduced and reliability increased.

The ability to detect and correct data errors is an important feature of any disk controller. The IMDC and DPLL support multiple-sector read/write with implied seek and automatic bad-sector handling. The IMDC can correct data errors by using either 32- or 40-bit user-specified ECC polynomials or with the CRC-CCITT polynomial.

One powerful IMDC feature is real-time record processing, which allows the user to specify the search criteria on a character-by-character basis and to specify the action that is to occur on completion of the search.

The IMDC supports the IBM 3740 and Sytem 34 media formats as well as soft- or hard-sector formats. The device handles both frequency-modulation (FM) and modified-frequency-modulation (MFM) data formats with data rates of as high as 2M bits/sec for FM and 10M bits/sec for MFM—regardless of the host processor's operating speed.

The introductory version of the chip set controls as many as four rigid and/or floppy drives. At present, these drives must have Shugart SA850, SA1000 or Seagate ST560 standard interfaces. Future changes by Signetics in the IMDC's mask-programmable microcode will let you use the chip with other interfaces, such as ESDI, ANSI and SMD.

TABLE—PORT DEFINITIONS

INPUT			OUTPUT	
BUS LINE	SIGNAL NAME	DEFINITION	SIGNAL NAME	DEFINITION
D0	INDEX	RIGID-DISK INDEX SIGNAL	HEAD1	HEAD SELECT 2^0
D1	READY	DISK READY	HEAD2	HEAD SELECT 2^1
D2	SEEKC	SEEK COMPLETE	HEAD4	HEAD SELECT 2^2
D3	TR0A	TRACK-ZERO FIRST SIGNAL	HEAD8	HEAD SELECT 2^3
D4	TR1A	TRACK-ZERO SECOND SIGNAL	HEAD16	HEAD SELECT 2^4
D5	WFA	FLOPPY WRITE-PROTECT	NOT USED	
D6	WFB	WRITE FAULT	STEP	HEAD STEP PULSE
D7	NOT USED		DIR	DIRECTION OF HEAD
D8	NOT USED		LWC	LOW WRITE CURRENT
D9	NOT USED		MOT	MOTOR ON
D10	NOT USED		PRECOM	PRECOMPENSATION ENABLE
D11	NOT USED		HDL	HEAD LOAD FOR FLOPPIES
D12	NOT USED		SEL0	DRIVE 0 SELECT
D13	NOT USED		SEL1	DRIVE 1 SELECT
D14	NOT USED		SEL2	DRIVE 2 SELECT
D15	NOT USED		SEL3	DRIVE 3 SELECT

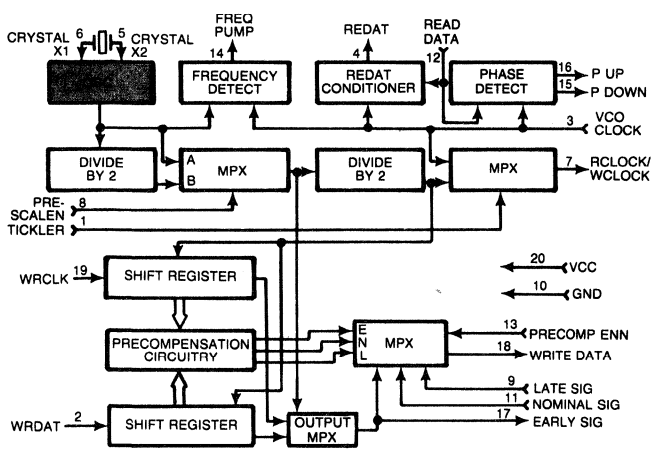


Fig 8—The Disk Phase-Locked Loop (DPLL) *uses an external voltage-controlled oscillator for the variable clock rate which tracks the read data from the disk unit.*

TABLE 2—ECA PARAMETERS USED BY EACH COMMAND

ECA PARAMETER USED	COMMAND MNEMONICS										
	WRMS	WRDD	VER	REMS	PRP	TSR	RETD	FORM	CALB	CORR	DIAG
COMMAND CODE	X	X	X	X	X	X	X	X	X	X	X
STATUS BYTES	X	X	X	X	X	X	X	X	X	X	X
MAX NUMBER OF RETRIES	X	X	X	X	X	X	X	X	X		
DMA COUNT	X	X	X	X	X	X	X	X		X	
COMMAND OPTIONS	X	X	X	X	X	X	X	X		X	
BUFFER ADDRESS	X	X	X	X	X	X	X	X		X	
BUFFER LENGTH REQUEST	X	X	X	X	X	X	X	X		X	
CYLINDER NUMBER	X	X	X	X	X	X	X	X	X		
HEAD NUMBER	X	X	X	X	X	X	X	X			
SECTOR NUMBER	X	X	X	X	X	X					
PRP COMMAND CONTROL WORD					X						
PRP PARAMETERS					X						
N0 - PREINDEX GAP								X			
N1 - POSTINDEX GAP								X			
N2 - SYNC BYTE COUNT	X	X						X			
N3 - POST-ID GAP	X	X						X			
N4 - POST-DATA GAP								X			
N5 - ADDRESS MARK CONTROL	X	X	X	X	X	X	X	X			
ECC MASK	X	X	X	X	X	X	X	X		X	
MOTOR-ON DELAY	F	F	F	F	F	F	F	F	F		
NUMBER OF HEADS	X	X	X	X	X	X	X	X			
ENDING SECTOR NUMBER	X	X	X	X	X	X	X	X			
STEPPING RATE	X	X	X	X	X	X	X	X	X		
HEAD SETTLING TIME	X	X	X	X	X	X	X	X	X		
HEAD LOAD TIME	F	F	F	F	F	F	F	F			
SEEK TYPE	X	X	X	X	X	X	X	X	X		
LOW WRITE CURRENT BOUNDARY TRACK	X	X	X	X	X	X	X	X			
PRECOMP BOUNDARY TRACK	X	X	X	X	X	X	X	X			
MAX NUMBER CYLINDERS/TRACKS	X	X	X	X	X	X	X	X			
SECTOR LENGTH/LAST SECTOR NUMBER	X	X	X	X	X	X	X	X			
FLAG BYTE			X	X		X	X				
B-TREE POINTER					X						

mable Record Processing (PRP) command), you can instruct the IMDC to locate a specific string of characters within a logical data block on a disk. After the match, the IMDC can perform a read of the logical data block, process a pointer within the data block or locate all blocks that satisfy some search criteria.

You specify the PRP functions by filling the Scan Control Word Table (SCWT) with the string and instruction parameters. The SCWT is part of the system memory, and its address is placed into the ECA table in the PRP parameters area. Each word of the SCWT consists of two parts, one data and the other an instruction; each part is eight bits long. The IMDC divides the FIFO memory into two parts for the PRP command; one part is dedicated to the character string and instructions, and the other is to be used during the disk data transfers. The FIFO memory receives the PRP commands and character string before the operation begins.

The physical sector address and a byte-count offset specify the start of the first byte to be scanned. You can specify the end of the record for which you are looking in terms of either a fixed record length or an end-of-record character.

You can transfer either the records for which you are searching or their locations to the system memory. In either case, you begin by reading records on the disk sequentially. When a record meets the search criteria, you can either immediately transfer the record's data into the system memory, or you can tell the IMDC to log the location of the record match and return the logged location to system memory.

You can write a software driver for the IMDC in two levels: entry level—for cold-start (power-on) entry to

the drives, or normal level—for access to the drives during normal operation. Both levels can be part of the same driver, with flags and tests to route the program to the correct procedures.

As previously noted, the IMDC registers, ECA pointer table and ECA blocks must be initialized before any operation is performed. A typical operating system would, at power up, put into the ECA pointer table the address of each ECA block and fill the ECA block with the parameters that define the disk types. Although these parameters would not usually change during normal operation, a diagnostic might call for such a change. **Table 2** shows the ECA block parameters used by each command.

During successive disk operations, the parameters most likely to change from one operation to the other, are the Command Code, Buffer Address and the Physical Disk address (cyclinder, head and sector), which define the what, where and how much of the requested transfer. Therefore, the operating system should, at least, pass these parameters to the normal-level software driver. In some cases, the buffer length will be different than the sector length, so it, too, should be updated with each command. For example, a read (REMS) operation without any data transfer becomes a seek command.

The last thing the driver must know is what drive is to be activated. This unit's number becomes the busy bit set in register 7 of the IMDC, reflecting the fact that it is the selected drive. As it is possible to request more than one operation at a time, the driver must be sure that it does not permit a previously requested drive's bit to be overwritten. **EDN**

Memory access control

Basic Memory Access Controller (BMAC)

Originally published by Signetics February 1985

Advance Information

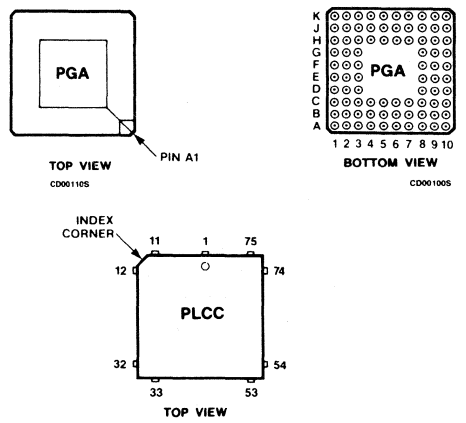
DESCRIPTION

The Signetics SCC68905 Basic Memory Access Controller (BMAC) is designed for the S68000 family (68000, 68010 and 68012) of microprocessors. It has a 24-bit logical address and a 24-bit physical address space. The BMAC mediates all accesses between the processor and system memory by controlling the memory hierarchy, translating virtual addresses into real addresses, supporting context switches and providing protection against unauthorized memory access. The operating system allocates the system memory and the BMAC implements these functions dynamically. The BMAC also provides support for a local memory.

FEATURES

- **Mediates memory reference between processor and memory**
- **Functions as a demand-driven MMU and a cache controller**
- **No wait states on cache hits**
- **MMU action occurs in parallel with cache action**
- **Little mutual bus interference with up to four processors**
- **Ensures correct data in required local memory**
- **Variable length segments with paging, variable length contiguous segments, or paged only**
- **Allows partitioning the virtual address space into one to four regions**
- **Selective flushing by regions**
- **Provides four protection types at the page or segment level**
- **Maintains four history bits**
- **Configurable cache block size of 2, 4, 8, or 16 bytes wide**
- **Configurable cache depth of 512, 1024, or 2048 bytes.**
- **Configurable page size of 1k, 2k, 4k, or 8k bytes.**
- **Can use sequential accesses to fill cache if available on the system.**

PIN CONFIGURATION



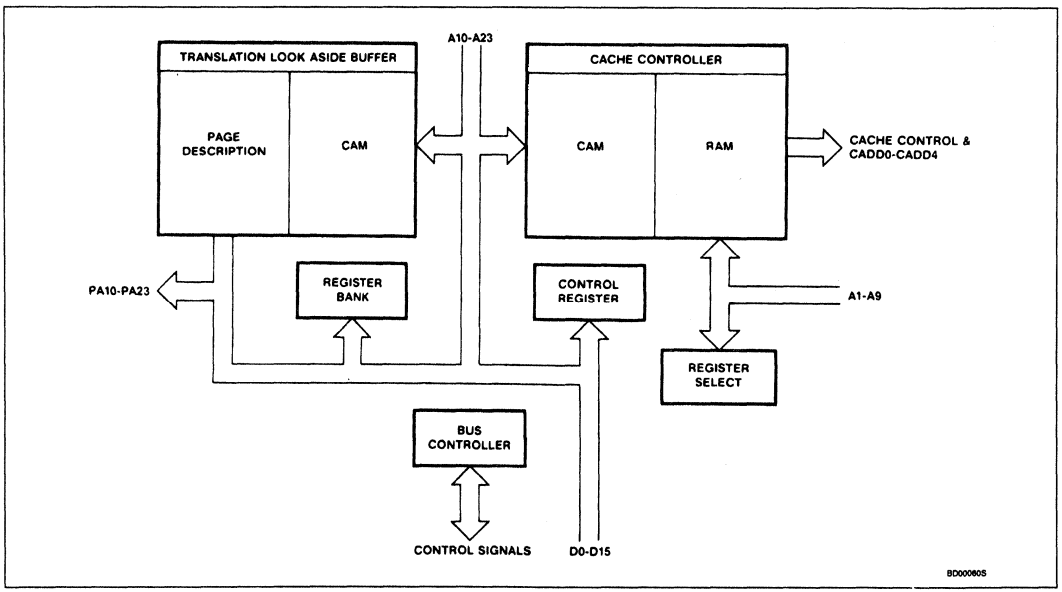
PGA	Function	PGA	Function	PLCC	Function	PLCC	Function
A-1	A3	F-1	V_{CC}	1	CADD3	43	MASN
A-2	A5	F-2	D15	2	CADD4	44	V_{SS}
A-3	A7	F-3	D14	3	A9	45	DTACKINN
A-4	A8	F-8	ASN	4	A8	46	CCEN
A-5	CADD3	F-9	BERRINN	5	A7	47	LACLR
A-6	CADD2	F-10	V_{CC}	6	A6	48	DTACKOUTN
A-7	CADD0	G-1	D13	7	A5	49	PA10
A-8	A12	G-2	D11	8	A4	50	PA11
A-9	PA23	G-3	D9	9	A3	51	PA12
A-10	A15	G-8	PA19	10	A2	52	PA13
B-1	FC2	G-9	PA20	11	A1	53	PA14
B-2	A1	G-10	CLOCK	12	FC0	54	PA15
B-3	A2	H-1	D12	13	FC1	55	PA16
B-4	A6	H-2	D7	14	FC2	56	PA17
B-5	CADD4	H-3	D5	15	FC3	57	PA18
B-6	CADD1	H-4	D1	16	RESETN	58	PA19
B-7	A10	H-5	PALE	17	CSN	59	PA20
B-8	PA22	H-6	DTACKINN	18	R/WN	60	CLOCK
B-9	A13	H-7	PA11	19	NC	61	BERRINN
B-10	A17	H-8	PA15	20	UDSN	62	ASN
C-1	FC3	H-9	PA16	21	LDSN	63	V_{CC}
C-2	FC1	H-10	PA18	22	V_{CC}	64	A23
C-3	FC0	J-1	D10	23	D15	65	A22
C-4	A4	J-2	D6	24	D14	66	A21
C-5	A9	J-3	D4	25	D13	67	A20
C-6	V_{SS}	J-4	D0	26	D12	68	A19
C-7	A11	J-5	BERROUTN	27	D11	69	A18
C-8	PA21	J-6	V_{SS}	28	D10	70	A17
C-9	A14	J-7	DTACKOUTN	29	D9	71	A16
C-10	A19	J-8	PA13	30	D8	72	A15
D-1	R/WN	J-9	PA14	31	D7	73	A14
D-2	CSN	J-10	PA17	32	D6	74	A13
D-3	RESETN	K-1	D8	33	D5	75	PA21
D-8	A16	K-2	D3	34	D4	76	PA22
D-9	A18	K-3	D2	35	D3	77	PA23
D-10	A20	K-4	LOCAL	36	D2	78	A12
E-1	LDSN	K-5	CWEN	37	D1	79	A11
E-2	NC	K-6	MASN	38	D0	80	A10
E-3	UDSN	K-7	CCEN	39	LOCAL	81	CADD0
E-8	A21	K-8	LACLR	40	BERROUTN	82	CADD1
E-9	A22	K-9	PA10	41	PALE	83	V_{SS}
E-10	A23	K-10	PA12	42	CWEN	84	CADD2

Advance Information

ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%, T_A = 0°C to 70°C
Ceramic PGA	SCC68905CCP84
Plastic LCC	SCC68905CCA84

BLOCK DIAGRAM



The BMAC provides the basis for a total memory management solution. The BMAC combines a memory management unit (MMU) and a cache controller in the same package. The MMU contains locations for 32 descriptors which, on demand, are loaded by the BMAC. Also, the BMAC contains hardware support to define the region context and boundaries. The cache controller provides the matching and control logic for a virtual cache which reduces memory access times by at least one clock cycle. With the BMAC, a cache hit will result in a no wait state access for even the fastest microprocessors.

The MMU and the cache are programmer configurable. The BMAC allows the system programmer to choose segments, pages or paged segments and the cache size is configurable in both width and depth.

The BMAC is functionally compatible with the MAC (SCC68910); updating the MMU is accomplished by software on the BMAC and by hardware on the MAC.

FUNCTIONAL DESCRIPTION

In most real modular systems, an 8MHz 68010 with a MMU needs between two and four wait states when it accesses memory. Speed and memory management pose problems for the system designer even for the 8MHz 68010 and it will be further aggravated when the higher speed 68010s and 68020s become available. This problem will be slightly reduced when the 256K DRAM is easily obtainable; but it is clear that a 16MHz 680xx and MMU will need at least 4 wait states when accessing memory unless that memory is not on a bus. The BMAC reduces this problem by functioning as a cache controller and a memory management unit. It performs access control of the memory hierarchy, translation of virtual to real addresses, supports context changes, and protects against unauthorized access during memory processing.

The operating system is responsible for allocating memory and access rights. The memory hierarchy can consist of three intermediate access store elements; a cache, a local memory, and the system memory. The cache memory element contains a copy of scattered sections of the local and system memories and is transparent to the software. The cache is a small, fast memory that allows the processor to see the cache access time instead of the usual system memory access time for 90% of its accesses and reduces the bus usage by 85%. The BMAC implements these functions dynamically.

Local memories can be attached to each processor in a multiprocessor system. The BMAC provides facilities to control the movement of code and data in and out of these local memories. This feature is transparent to the user software.

A demand paged virtual memory system can be implemented with one or more BMACs. The BMAC makes a multiprocessor system practical by reducing bus utilization. The BMAC has a special provision that allows multiple processing or processors to operate even if they share common data. If an 85% reduction in bus usage is not adequate, a local memory can then be attached to each processor which will reduce it further.

Advance Information

PIN DESCRIPTION

Signal names ending in 'N' are active low. All other signals are active high. In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' and 'negated' independent of whether the signal is active in the high (logic one) state or the low (logic 0) state. Refer to the individual pin descriptions for the definition of the active level of each signal.

MNEMONIC	PIN NO.		TYPE	DESCRIPTION
	PLCC	PGA		
A23 – 10	64 – 74, 78 – 80	E-10 – E-8 D-10,C-10, D-9,B-10, D-8,A-10, C-9,B-9, A-8,C-7, B-7	I	**Address Lines:** Active high. High order virtual address lines issued by the processor.
A9 – 1	3 – 11	C-5,A-4, A-3,B-4, A-2,C-4, A-1,B-3, B-2	I	**Address Lines:** Active high. These lines are the page address or when CSN is asserted, the BMAC register address. These are I/O lines on the MAC.
PA23 – 10	77 – 75, 59 – 49	A-9,B-8, C-8,G-9, G-8,H-10, J-10,H-9, H-8,J-9, J-8,K-10, H-7,K-9	O	**Physical Address:** Active high, three statable. Some or all of pins PA12-10 may always be tri-state on the BMAC, depending on the page size programmed in the PDR. This is done to maintain campatibility with the MAC.
CADD4 – 0	2, 1, 84, 82, 81	B-5,A-5, A-6,B-6, A-7	O	**Cache Address:** Active high. High order cache address lines.
D15 – 0	23 – 38	F-2,F-3, G-1,H-1, G-2,J-1 G-3,K-1, H-2,J-2, H-3,J-3, K-2,K-3, H-4,J-4	I/O	**Data Lines:** Active high, three-statable. These lines are used to transfer data between the processor and the BMAC when the CSN line is asserted.
FC3 – 0	15 – 12	C-1,B-I, C-2,C-3	I	**Function Codes:** Active high. These lines are issued by the processor.
ASN	62	F-8	I	**Address Strobe:** Active low. Issued by processor.
MASN	43	K-6	O	**Memory Address Strobe:** Active low. To memory issued by BMAC.
UDSN	20	E-3	I	**Upper Data Strobe:** Active low, three-statable. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
LDSN	21	E-1	I	**Lower Data Strobe:** Active low, three-statable. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
DTACKINN	45	H-6	I	**Data Transfer Acknowledge In:** Active low. Data acknowledge input from memory.
DTACKOUTN	48	J-7	O	**Data Transfer Acknowledge Out:** Active low, open-drain. Data acknowledge output to processor.
R/WN	18	D-1	I	**Read/Write:** Active high for read, low for write. This signal is an I/O line on the MAC. This signal line requires an external tri-state buffer since the 680xx will not make this line high impedant when the BMAC is accessing memory.
CSN	17	D-2	I	**Chip Select:** Active low.
CCEN	46	K-7	O	**Cache Chip Enable:** Active low. This signal has to be qualified externally by the data strobe signals.
CWEN	42	K-5	O	**Cache Write Enable:** Active low.

Advance Information

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	DESCRIPTION
	PLCC	PGA		
LOCAL	39	K-4	O	**Local:** Active high, three-statable. Local memory enable.
BERRINN	61	F-9	I	**Bus Error In:** Active low. Bus error signal, input from bus. During a memory access, DTACKINN and BERRINN cannot both be asserted unless BERRINN is asserted two clock cycles before DTACKINN.
BERROUTN	40	J-5	O	**Bus Error Out:** Active low, open-drain. Bus error signal, output to processor.
CLOCK	60	G-10	I	**Clock:** Active high. Clock input.
RESETN	16	D-3	I	**Reset:** Active low. When asserted this signal will reset the BMAC.
PALE	41	H-5	O	**Physical Address Latch Enable:** Active high. When high, it enables the counter to be incremented on a block fetch.
LACLR	47	K-8	O	**Latch Clear:** Active high. This line is pulsed low to clear the counter for a block fetch.
V_{CC}	22,63	F-1,F-10	I	+5volt ±5% power input.
V_{SS}	83,44	C-6,J-6	I	Power ground return.

MEMORY STRUCTURE

The BMAC uses a memory which is divided into variable length segments, which in turn can be divided into fixed length pages. Even if pages are not used, the BMAC measures the length of the segment in quantities of 1, 2, 4, or 8k bytes. As any segment can be any number of pages in length, the system can work if desired with only two segments - program and data. Protection is provided on a segment basis or page basis. Segments, or pages, can have the following protection attributes:

1. Supervisor permission
2. Read permission
3. Write permission
4. Execute permission

These attributes are recognized and protected dynamically by the hardware. If a user attempts to access a segment without permission, a bus error (BERR) signal is generated.

Aside from protection attributes, the BMAC supports three additional attributes which are used to control the memory hierarchy.

1. Non-cacheable. This attribute forces associated segments or pages to never be cached. It forces segments or pages which are shared between processes to have only one copy which is always up to date.
2. Local. If the system has local memory then this attribute is used to force the associated page into the memory local to the processor and saves bus accesses.
3. Contiguous. This attribute, if set, indicates a segment that has no page table associated with it. Such segments are to be loaded and relocated in memory as a whole.

CACHE

A cache contains a copy of portions of program and data which are in the memory. Its associated principle works because most words of memory are accessed more than once within a relatively short time interval. The BMAC automatically stores each word read by the processor in the cache and, if the processor tries to read that word a second time, the BMAC provides the word from the cache instead of accessing the memory. When the processor writes a word, the BMAC implements a write-through policy, i.e., the word is written into both the cache and the memory. If a segment is marked as local, then this refers to the local memory and not the system memory. If the bus is not available to the processor, the BMAC allows the processor to proceed, and if the bus is granted before the next 'cache miss' or before the next write, the processor need not be aware that there was a bus access latency time.

Under certain conditions, e.g., when two processors are sharing access to a common segment of memory interleaved in time, it may not be desirable to cache the data. These segments can be marked as non-cacheable.

Semi-Associative Caches

Semi-associative caches are attempts to temper the idea of an associative cache with a dose of reality. Associative caches have an identifier attached to each cache word. The identifier is a content addressable memory (CAM) which holds the high order bits of the address. Basically, a semi-associative cache adds one assumption to the idea of associative caches. The assumption is that there is a working set of memory locations which in fact is reasonably small and tend to cluster.

In a semi-associative cache, the cache itself is divided into a number of sub-caches, which is called the set size. These sub-caches are defined by the high order address bits as shown by the following example. An address n bits in length is divided into three areas:

L bits (label) – high order address bits which define the sub-cache

I bits (index) – used to access the sub-cache (the algorithm will be given later)

B bits (offset) – used only if the block size is greater than the word size. The block size is the width of the data path between the cache memory and the memory. It can be wider than the processor's data bus.

ADDRESS FORMAT

Label (L Bits)	Index (I Bits)	Offset (B Bits)

The label field defines a sub-address space of $2^{(I+B)}$ words in length. Up to S (set size) of these address spaces can be mapped into the S-subdivisions of the cache simultaneously. Note that the mapping between one of these sub-address spaces and a sub-cache is not fixed but varies dynamically.

The first step in checking if a particular word is in the cache is to see if its sub-address space has been mapped into a sub-cache. If this is the case, then the next step is to see if the particular word is present. Thus, a bit-per-word presence is needed. When a word is read into the cache for the first time the presence bit is set. Subsequently, when the word is accessed, the cache controller checks to see if the sub-address space has

Advance Information

been mapped into a sub-cache and that the presence bit is set. If both are affirmative, the word is accessed out of the cache. Note that the path to memory can be wider than the path to the processor. In this case, a block which is the width of the path to memory is the unit of access (the presence bit represents a block not a word); the B offset bits are not used in the cache controller but are used externally to select the word from the block. If a block is to be read into the cache via multiple reads, this is controlled by the BMAC.

The basic algorithm to see if a word is present in the cache is :

1. Compare the label field against a CAM of length S and width L. If there is a match go on. If not, a miss has been recorded and go to step 3.
2. Each sub-cache has a 2^l x 1 RAM associated with it in order to see if the desired block (word) is present, i.e., each block has a presence bit associated with it. Use the index to access this RAM. If the presence bit is set, then the required data is present and the B offset bits can be used to access it. If the presence bit is not set, then cause a memory access or memory accesses, and load the data into the cache setting the relevant RAM bit.
3. This step is taken only if the address space denoted by the label has no sub-cache allocated to it. Therefore a sub-cache needs to be allocated. The oldest allocated sub-cache is freed, i.e., a label sub-space is de-allocated and the needed label is allocated to the sub-cache. All the presence bits of the sub-cache are reset to zero and step 2 is implemented. In this case a new mapping has to be made between the sub-caches and the addresses. Note that steps 1 and 2 can be taken in parallel, with the output of step 1 being used to validate the output of step 2.

Note that this replacement algorithm is essentially a FIFO mechanism. Simulations have shown that, with a working set the size of the BMAC's, there is no real performance difference between a FIFO and a replacement algorithm.

VIRTUAL MEMORY

The view of virtual memory presented here is supported by a mixture of hardware and software. It may well appear to be too complex for simple implementations, or not complex enough for high security systems. In the former case, the system can be simplified. In the latter case, the hardware protection can be extended by software to increase the degree by subtlet or range of the protection mechanisms. These software extensions are protected by hardware so that they cannot be circumvented. A virtual address can be split into three portions.

1. Segment number
2. Page number
3. Offset within a page

This split is supported by hardware. The split between 1 and 2 is a convention of the software established at SYSGEN time, and the BMAC can be programmed accordingly. The choice of the split between 2 and 3 is more restricted and is also handled by the BMAC.

Partitioning of the Address Space

The virtual address space, as seen by the processor and the software, can be partitioned into up to four different regions each of which has its own segment table. This partitioning need not occur, i.e., the address space can be left as one region. These regions which are configured by software are provided for a fast and protected operating system.

Virtual Address Space Regions

Four regions are defined sequentially starting from the bottom end of the virtual address space. The upper limit of each region is specified in turn. This implicitly defines the lower limit of the next region. Thus if only one region is to be specified, then its upper limit is given as FFFFFFFF. The four regions are all equivalent.

Note that the virtual address space can be defined by the address only. The function codes are then used for protection. The concept of address spaces in the 68451 MMU is replaced here by the concepts of the regions. The latter are used in a similar but less restricted fashion than the former to enable different processes each to have its own unique virtual address space. Naturally portions of this address space can be shared with other processes.

Each region on the MAC has its own segment table defined implicitly by the segment table pointer registers (STPRs) and segment table length registers (STLRs). On the BMAC, these registers are emulated. The segment table defines which areas of the virtual address space are accessible to the current process. The contents of a region are flushed from the cache by resetting the associated STPR on the BMAC. This can only be done in the supervisor mode. Note that the BMAC does not use the STLRs, and it uses the STPRs only to invalidate the caches. Region definition is done via the RDR register.

Use of the Regions in a Simple System

A simple system is defined as having only one region which is accessible by all function codes. Thus protection is performed by the protection bits associated with each segment or page. This system is configured by writing FFFFFFFF into all region definition registers, thus defining the upper boundary of R0 to be FFFFFFFF and the lower boundary of R0 to be 00000000. The system can be used as if it had only one region which stretches uniformly from 0 to FFFFFFFF.

Full Use of the Regions

This section gives an example of how all four regions can be used in the implementation of an efficient operating system which protects users from one another and from itself. Region R0 is used only by the privileged part of the operating system and contains the part of the kernel which is privileged. Thus, all segments in this region are marked with an indication that supervisor permission is needed to access them. This part of the operating system should be as small as possible, because it has access to all of memory.

The next region, R1, is used by the supervisor. This is the part of the operating system that provides the non-privileged functions. This code is nevertheless 'trusted code' and can have access to resources that are not available to user programs. On a system call, the handling routine in R0, not only switches the call to the relevant routine in R1 but also disables the privileged state and enables R1.

There is often a large body of utilities in a system which is used by most programs. If the data which they access is protected, then there is no reason why these utilities cannot automatically be made available to all users in an execute only mode. Thus, R2 is used as a utility region available to all processes and always enabled.

The last region, R3, is the user region. This region contains the process specific code and data. The mapping of this virtual address space into an associated physical address differs from process to process. This remapping is performed by changing registers in the BMAC.

DATA STRUCTURES

The data structures are not necessary for the BMAC but logically exist within an operating system, as the BMAC does not directly access these tables. They are noted here in order to ensure compatibility with the MAC. The operating system clearly needs information as to where its components are. In this example, this will be called the system block (SB). Each process or task also needs a set of parameters which define its state. Although these parameters need not be gathered into a block in one place, we will assume that this has been done and they are present in the task control block (TCB).

The table structures involved in virtual memory are the segment tables and the page tables. The TCB contains pointers to and length registers for the two segment tables currently in use by the process. That segment table contains entries describing the segments available to the process. A segment is a logical extent of memory which is to be treated in a uniform way, i.e., it has the same protection and location characteristics. The page table contains the address space mappings between virtual and real addresses obtained at that time. The page tables also contain page status, i.e., history and current place in the memory hierarchy. Pictorially this can be represented for an individual task as shown in figure 1.

This structure of a segmented memory with a paging scheme underneath has the following features:

1. Allows simple systems to be made.
2. Allows secure, protected systems to be made.
3. Allows sharing of segments with different permissions.
4. Allows the memory management software to keep track of which pages to swap.

The System Status Block

If any system regions are defined, then this block contains the pointers to the segment tables of the regions dedicated to the system. Normally these pointers are not changed, i.e., they are set on system load and are static thereafter. Note that these registers need not exist in memory because they are contained on the BMAC.

The Task Control Block

This must now contain a pointer to the process's segment table and a count of its current length. The count is necessary to ensure that a process cannot access an area beyond its segment table assuming that it contains a valid segment descriptor. When loading a new segment table pointer into the BMAC, all cache references in that region are automatically marked as invalid. The BMAC performs no other actions. For MAC compatibility, note that the STPR and STLR registers are in memory and are used by software not by the BMAC.

Segment Tables

At any instance, there are 1 to 4 valid segment tables depending on how many regions have been defined and enabled. The segment table consists of three fields.

Page table pointer – This 32-bit field contains the pointer to the page table which defines the current segment. Page tables must be aligned on long-word boundaries in memory.

Page table length – This 22-bit field contains the page number of the last page in the segment.

Segment protection – This 10-bit field contains the bits which define the status of the segment. Each use of the segment is protected differently depending on the permissions granted to each process. These bits are:

1. Valid bit (V). This bit is used to indicate whether the segment is valid or not.
2. Supervisor permission (S). If this bit is set, supervisor permission is necessary to access this segment.
3. Access permission bits (R, W, E). If these bits are set, corresponding access is available for the segment: read access (R), write access (W), execute access (E).
4. The following three bits define the mapping of the segment:
   a. Non-Cacheable (NC). If this bit is set, the associated segment will not be brought into the cache memory. For example, if a memory is used concurrently by the processors or memory addresses are used for I/O, this segment should be made non-cacheable.
   b. Local (L). If this bit is set, pages of the associated segment may belong to the local memory attached to the processor. It enables the use of local memory identification (LI) field.
   c. Contiguous (C). If this bit is set, the associated segment does not have a page table. It is segment-only and should be mapped into memory as a whole.
5. Software protection (SP). This bit is not used by the BMAC but may be used by the software.
6. Offset bit (O). This bit is used to denote whether pages of a segment start at the top (higher address, O = 1) or bottom (lower address, O = 0) of the segment's virtual address space.

For a contiguous segment configuration, additional bits are defined in the second word. These bits are defined for the MAC and should be set to 0000 for the BMAC to maintain compatibility.

Figure 2 shows the format of the segment descriptors in the segment table for a paged (non-contiguous) segment and figure 3 shows the format of the contiguous segments' descriptors.

Page Tables

The page table is split into two fields totalling four bytes which are always used by the hardware. They contain:

1. The 18-bits pointer to the physical address of the beginning of the page.
2. Three page-history bits:
   - Present (P): page is present in memory.
   - Used (U): page is used since this bit is last reset.
   - Dirty (D): page is written to since this bit is last reset.
3. Two reserved bits (Rs). These bits may not be used by software for any purpose.
4. Local memory identification (LI). The BMAC supports systems with up to 15 local memories. If a segment is marked local, those bits indicate in which local memory the page is resident. This will be checked against the contents of the processor identification register. This check should be done by the BERR routine which loads the descriptor into the SCC68905. If LI = 1111, then the page is considered to reside in system memory.
5. A non-cacheable (NC) attribute bit. If this bit is set, the associated segment will not be brought into the cache memory. For example, if a memory is used concurrently by the processors or memory addresses are used for I/O, this segment should be made non-cacheable.

In addition, are the page protection bits. These bits are:

1. Supervisor permission (S) needed to access this page
2. Three access permission bits. Read access (R), write access (W), execute access (E). The three access permissions bits allow the access mode if set.

Figure 4 shows the format of the page descriptors in the page table in memory.

Residence of System Tables

It may be desirable to place most of the system tables mentioned above in virtual memory without necessarily having them in physical memory all the time.

Memory Protection and Sharing

The BMAC provides mechanisms for implementing systems with varying degrees of protection. The simplest protected system would only separate user code from operating system code. Protection is basically provided only by giving a process access to the resources it needs. These resources include memory and I/O. Access to I/O is accomplished by giving a process access to the necessary I/O management routines. This access can be execute-only and the entrance to these routines can be controlled via hardware in the normal system call manner or by putting a software 'gateway' as the access point. This gateway is essentially an execute-only segment and/or page which can be marked so as to cause a trap when entered. Thus, a trusted program can be used to do the I/O access or to provide extra entry points to the calling program. For example, if

Advance Information

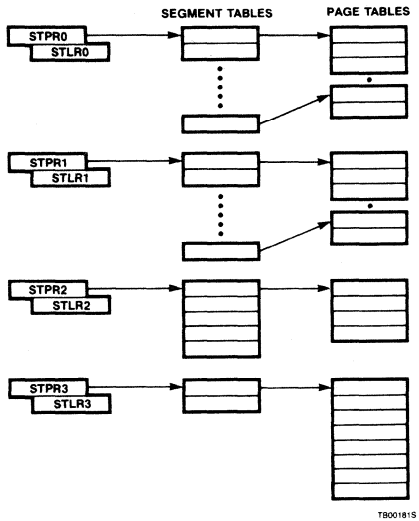


Figure 1. Table Structure

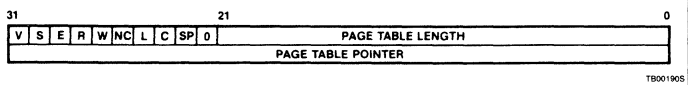


Figure 2. Segment Descriptor for a Paged Segment (C = 0)

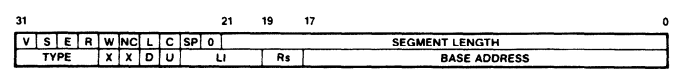


Figure 3. Segment Descriptor for Contiguous Segment

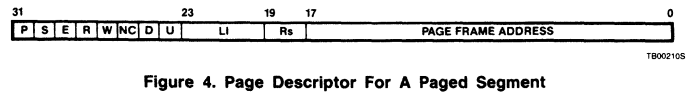


Figure 4. Page Descriptor For A Paged Segment

a file is opened then this technique can add segments to the calling process's virtual address space, where each segment performs an I/O function. The process will only have access to these routines.

This process will only be effective if a process is forbidden to access its segment and/or page descriptors. This is achieved by not putting the segment and/or page tables belonging to a process in a segment and/or page directly accessible to the process. This segment and/or page is accessible to the operating system.

Sharing of code and data can be accomplished by merely putting a segment and/or page descriptor pointing to the same page table in more than one process's segment and/or page table. Note that these segment and/or pages can have different virtual addresses for the different process. Message passing can then be easily and efficiently implemented by asking the operating system to 'add' a descriptor to another process's address space.

Sub-Setting Facilities for Simple Systems

The BMAC can be used as an MMU either with or without using the cache controller function. If a simple system is desired, a subset of the BMAC's capabilities can be used. The following are some examples of possible system configurations:

1. A purely paged system
2. A minimally protected system. This would have one region with two segments for code and data, with no paging within each segment.
3. A minimally protected stack-based system. This would have one more segment for each stack used to be used to easily allocate stack space as it is needed, while still controlling unlimited growth.

These examples are provided to emphasize that all the facilities need not always be used.

REGISTERS

The registers on the BMAC are used for three purposes.

1. By the processor to inform the BMAC of the hardware system configuration.
2. To force the BMAC to invalidate the cache descriptor entries.
3. By the BMAC to inform the processor of its status.

The BMAC will issue a BERR if chip select is asserted without using a supervisor function code. Registers on the BMAC can be accessed by the processor when the chip select input is active and address line A9 = 0. The register addresses are then decoded from address inputs A6–1. When chip select is active while A9 = 1, the BMAC will only check whether FC2 is asserted and issue a BERR if this is not the case.

Page Definition Register (PDR)

Only bits 0 – 1 of this register are used. These bits indicate the size of the page. Note that depending on the page size, some physical address pins of the BMAC will always be in tri-state mode. This is necessary for pin compatibility with the MAC. The pins that are tri-stated are listed in the following.

Bits 1 0	Page Size (Bytes)	Tri-stated Lines
0 0	1024	–
0 1	2048	PA10
1 0	4096	PA10-11
1 1	8192	PA10-12

Size of Segment Register (SSR)

Only bits 0 – 3 are used. This register, which is reset to zero on a BMAC reset, defines the

part of the address that selects a segment. The contents of the SSR is the encoded number of address bits (starting with A23) that are part of the segment number. The segment size can vary between 8k and 16M bytes in size.

If less than five address bits are used for the segment number, the split of the address space into regions is no longer supported. In those cases the whole address space will be treated as region 0.

Size of Cache Register (SCR)

Only bits 0 – 3 are used. This register specifies the cache organization. The meaning of the bits which are all reset to zero on a BMAC reset are:

Bits 3 – 0	Block Size	Cache Depth
0000	16 bits	512
0001	32 bits	512
0010	64 bits	512
0011	128 bits	512
0100	16 bits	1024
0101	32 bits	1024
0110	64 bits	1024
0111	128 bits	1024
1000	16 bits	2048
1001	32 bits	2048
1010	64 bits	2048
1011	128 bits	2048
11xx	Reserved	

Note that bits 2, 3 define the cache depth and bits 0, 1 define the cache block size.

Region Definition Register (RDR)

This register defines the boundaries of regions 0, 1 and 2 explicitly and region 3 implicitly. The register is split up into three fields which define the five most significant bits of the address for each region. These bits are then extended to a full address with 19 1's in the least significant bit positions. Bits 0 – 4 define the highest address in region 0. Region 1 is defined from the top of region 0 to the address specified by bits 5 – 9 of the RDR, extended with 19 1's. Similarly, region 2 is defined by the top of region 1 at its low end and bits 10 – 14 of RDR at its high end. Region 3 is defined from the top of region 2 to FFFFFF at its top end. Bit 15 is not used.

Region Attribute Register (RATR)

In order to remain compatible with the MAC, the BMAC will respond with a DTACK when this register is addressed. This is the only action the BMAC will perform when this register is accessed. On the MAC, this register defines the region attributes.

Processor Identification Register (PIR)

Only bits 0 – 3 are used. The number encoded in the four bits of this register identifies one of the 15 processors. Code 1111 in this register is reserved.

BMAC Mode Register (MMR)

Only bits 0 – 1 are used. On reset, the BMAC resets this register to zero. It has to be set if the BMAC is to be used. When the BMAC is disabled, the cache is effectively not present and all addresses are passed through the BMAC without any change. When the BMAC is enabled, the addresses are translated in the chosen manner and the cache is used in the configuration given in the SCR. If bit 0 is set, it indicates that the cache memory is enabled and if bit 1 is set, it enables the BMAC.

Note that loading this register will enable the BMAC. This register should only be loaded when all the other registers have been loaded.

Memory Fetch Register (MFR)

Only bits 0 – 1 are used. This register indicates how many memory accesses are needed when the BMAC loads a cache block.

Bit 1	Bit 0	No. Memory Accesses
0	0	1
0	1	2
1	0	4
1	1	8

The number of memory accesses also corresponds to the block size.

Segment Table Pointer Registers (STPR0 – 3)

In order to remain compatible with the MAC, the BMAC will respond with a DTACK when any of these registers is addressed. The STPR registers are 32-bits wide; they are implemented by two 16-bit registers, upper and lower word. When the STPRnLs are loaded, all the mappings in the BMAC in the relevant region are reset, i.e., the cache appears empty of all addresses in the region and all page descriptors in the BMAC in the region's address space are invalidated.

Segment Table Length Registers (STLR0 – 3)

In order to remain compatible with the MAC, the BMAC will respond with a DTACK when any of these registers is addressed. On the MAC, these registers contain the length (number of segments) of the segment tables currently valid.

MAC Communication Area Pointer Register (MCAPRU, MCAPRL)

Logically this register is 32 bits; it can be accessed by the upper and lower word. In order to remain compatible with the MAC, the BMAC will respond with a DTACK when this register is addressed. On the MAC, this register contains the physical address of the MAC communication area in memory.

Translate Command Register (TR)

In order to remain compatible with the MAC, the BMAC will respond with a DTACK when this register is addressed. On the MAC, a write to this register will cause the MAC to execute the translate command.

Error Code Register (ECR)

This register contains the reason for the bus error. The error code is set when BERROUT is asserted by the BMAC and when register DESCR1 is loaded. The meaning of the bits is:

bit 0, 1 Contain the region number where the error occurred.

bits 2 – 6 5-bit error code. The following error causes are encoded for the BMAC. All other codes are reserved for future use.

		EC			
4	**3**	**2**	**1**	**0**	
0	0	0	0	0	No error
0	0	0	0	1	Page not in required local memory
0	0	0	1	0	Access permission violation
0	1	0	0	0	Illegal BMAC access
0	1	1	1	1	BERR on global bus

An access permission violation occurs when the current bus cycle does not have the access permissions required for the segment or page that it attempts to access. The access permissions of a page are S (supervisor permission needed), E (execute access), R (read access) and W (write access). These four bits are stored for each page in the page translation look-aside buffer (TLB) on the BMAC and they represent the exclusive OR of the corresponding bits in the segment and the page tables.

The current bus cycle is characterized by the signals FC3 or (processor/I/O cycle), FC2 (supervisor/user mode), FC1 (program/data access) and R/W (read/write). Access permissions will not be checked during an I/O cycle (FC3 = 0) and during a CPU-space

cycle (FC3-0 = 1111). Table 1 shows the required access rights (S, E, R, W) for all

Advance Information

Table 1. ACCESS RIGHTS

FC3 0: I/O 1: PROC	FC2 USER SUPERV	FC1 DATA PROGRAM	FC0	R/W WRITE READ	S	E	R	W
1	0	0	X	0	0	X	X	1
1	0	0	X	1	0	X	1	X
1	0	1	X	0	N	N	N	N
1	0	1	X	1	0	1	X	X
1	1	0	X	0	X	X	X	1
1	1	0	X	1	X	X	1	X
1	1	1	0	0	N	N	N	N
1	1	1	0	1	X	1	X	X
1	1	1	1	X	X	X	X	X
0	X	X	X	X	X	X	X	X

X = do not care, N = violation, illegal input combination.

possible non CPU-space bus cycles. All other combinations will result in an access permission violation.

bit 7 Valid only when bit 15 is set. If ECR(15) = 1, this bit, if set, indicates a page descriptor miss; if not set, it means the dirty bit is not set on a write.

bits 8 - 10 Not used, reserved.

bits 11 - 14 S, E, R and W access permission bits of the currently accessed page. This field is valid only when an access permission violation is reported.

bit 15 If set, it indicates a TLB miss; to be handled by the BMAC TLB loading software routine. If not set, an access error as specified by the error code has occurred.

Test Mode Register (TMR)

This is an 8-bit register that can be read and written into. The meaning of the bits is as follows:

bit 0 If set, this bit will cause the signal SCLOAD and PDLOAD to be generated whenever and as long as ASN is active. This will allow for efficient writing to the CAMs.

bit 1 If set, this bit selects:
A31–5 as input for address CAM. (normal input)
A4 as input for flush CAMs. (multiplexed input)
A3 as input for dirty CAM. (multiplexed input)
A2,A1 as input for region CAMs. (normal input)
FC3 - 0 as input for permission CAMs. (multiplexed input)
HALLC as input for CADD0 output buffer (multiplexed output)
MALLC as input for CADD1 output buffer (multiplexed output)
HALLP as input for CADD2 output buffer (multiplexed output)
MALLP as input for CADD3 output buffer (multiplexed output)
This bit must be set during the entire CAM test sequence.

bit 2 If set, this bit selects THIT as input for PALE output buffer. It will also generate SPRBIT whenever and as long as CS is active. This bit must be set during the present-bit RAM testing. It allows for writing to the present-bit RAM.

bit 3 If set, this bit will disable the automatic resetting of present-bit RAM column on an SCLOAD.

bit 4 If set, this bit selects NC as input for LACLR output buffer. This bit must be set during the PD RAM testing.

bit 5 If set, this bit selects DECTST1 as input for CADD4 output buffer and DECTST0 as input for CWEN output buffer.
This bit must be set during register/command decoder test.

bit 6 Reserved

bit 7 Reserved

Descriptor Base Register (DBR)

Only bits 0–1 are used. This 2-bit register contains the next slot to be used after slot 31 in the TLB.

Descriptor Registers (DESCR0 - 2)

These registers are used to load descriptors. The BMAC requires the most significant 16 bits of the first long word of the segment descriptor to be written to register DESCR0, that the most significant 16 bits of the page descriptor be written to register DESCR1, and that the least 16 significant bits of the page descriptor be written to register DESCR2. Registers DESCR0, DESCR1, and DESCR2 must be written in that order. It is recommended that the user use the MOVEM instruction to transfer the data to the registers.

Register Summary

Tables 2 and 3 describe the register map and bit map formats.

BMAC OPERATION

The BMAC can be thought of as two cooperating units; the cache controller section and the memory mapping unit (MMU). Between them, all of the BMAC functions are provided.

The Cache Controller

The cache controller manages a semi-associative cache. It has a working set of 32 sub-caches. These sub-caches are defined not only by the address, but also implicitly by the address space. The BMAC recognizes four address spaces defined by the regions. The region definitions are independent of their use in the system. System designers can use them for whatever purpose is required. However, the same virtual address cannot be used in different regions. The distinction between the regions, in the cache controller, is made purely to allow the cache entries belonging to the separate address spaces to be invalidated for a region as a whole. If a system's dynamic working set is favorable, this will allow interrupt service routines, for instance, to remain in the cache while the processor continues a process or even changes processes.

On a reset, the cache is marked as empty and all 32 sub-caches are available to all four regions. All sub-caches which are in use by a region are automatically marked as empty whenever that region's segment table pointer register is loaded. Accesses to the CPU address space function code 1111 are never cached; they are passed through to the physical address bus. Upon a CPU space access (FC3 - 0 = 1111 and ASN are asserted), the BMAC will pass the virtual address on to the physical address pins and assert MAS. When the DTACKIN (or BERRIN) becomes active, the BMAC will assert DTACKOUT (or BERROUT). The cycle will be finished when ASN is negated by the processor.

A word can only get into the cache if it has satisfied the criteria of the segment descriptor. Thus, when a word is read and it is found in the cache, it is valid if the processor has the necessary permissions. A copy of the segment permission rights is kept with the associated sub-cache. If a write is performed, then this is only written into the cache if the processor has write permission. If it is so desired, the cache can be automatically disabled on reset. It can be enabled by writing into the MAC mode register.

Table 2. BMAC REGISTER MAP

ADDRESS[1] 6 5 4 3 2 1	ACRONYM	REGISTER NAME	MODE	AFFECTED BY RESET	NOTES
0 0 0 0 0 0	PDR	Page definition register	R/W	Yes	
0 0 0 0 0 1	SSR	Size of segment register	R/W	Yes	
0 0 0 0 1 0	SCR	Size of cache register	R/W	Yes	
0 0 0 0 1 1	ECR	Error code register	R	Yes	
0 0 0 1 0 0	TR	Translate command register	W	No	3
0 0 0 1 0 1	PIR	Processor indentification register	R/W	Yes	
0 0 0 1 1 0	MMR	MAC mode register	R/W	Yes	
0 0 0 1 1 1	MFR	Memory fetch register	R/W	Yes	
0 0 1 0 0 0	TMR	Test mode register	R/W	Yes	
0 0 1 0 0 1	DBR	Descriptor base register	R/W	Yes	
0 0 1 0 1 0	DESCR0	Descriptor register 0	W	No	
0 0 1 0 1 1		Reserved			
0 0 1 1 0 0	DESCR1	Descriptor register 1	W	No	
0 0 1 1 0 1	DESCR2	Descriptor register 2	W	No	
0 0 1 1 1 0		Reserved			
0 0 1 1 1 1		Reserved			
0 1 0 0 0 0	STPR0U	Segment table pointer 0 upper	R/W	No	2, 3
0 1 0 0 0 1	STPR0L	Segment table pointer 0 lower	R/W	No	2
0 1 0 0 1 0	STPR1U	Segment table pointer 1 upper	R/W	No	2, 3
0 1 0 0 1 1	STPR1L	Segment table pointer 1 lower	R/W	No	2
0 1 0 1 0 0	STPR2U	Segment table pointer 2 upper	R/W	No	2, 3
0 1 0 1 0 1	STPR2L	Segment table pointer 2 lower	R/W	No	2
0 1 0 1 1 0	STPR3U	Segment table pointer 3 upper	R/W	No	2, 3
0 1 0 1 1 1	STPR3L	Segment table pointer 3 lower	R/W	No	2
0 1 1 0 0 0	RATR	Region attribute register	R/W	No	3
0 1 1 0 0 1	RDR	Region definition register	R/W	Yes	
0 1 1 0 1 0	Reserved				
0 1 1 0 1 1	STLR0	Segment length register 0	R/W	No	3
0 1 1 1 0 0	Reserved				
0 1 1 1 0 1	STLR1	Segment length register 1	R/W	No	3
0 1 1 1 1 0	Reserved				
0 1 1 1 1 1	STLR2	Segment length register 2	R/W	No	3
1 0 0 0 0 0	Reserved				
0 1 0 0 0 1	STLR3	Segment length register 3	R/W	No	3
1 0 0 0 1 0	MCARU	MAC communication area pointer upper	R/W	No	3
1 0 0 0 1 1	MCARL	MAC communication area pointer lower	R/W	No	3
1 0 0 1 0 0		Reserved			
•					
•					
•					
1 1 1 1 1 1					

NOTES:

1. All registers are accessible as 16 word bits only. Attempt to access these registers as one 8 bit byte will result in an illegal access error.
2. Access to STPR0 – 3 should be done in two consecutive word accesses or a long word access. The BMAC resets all mappings in the relevant region only when the lower word of the STPR0 – 3 is written to.
3. These registers maintain compatibility with the MAC. Access to these registers in the BMAC will be responded to only with DTACK. No other actions will take place.

If a read miss occurs, the BMAC will fetch enough words to fill the associated cache block. It does so with the aid of an external counter (74F193), and is connected as shown in figure 5 with associated timing in figure 6.

The physical address latch enable (PALE) is normally low and LACLR is also low so that the signals A3–1 pass through the 74F193. If a block is to be loaded, LACLR is raised high for one clock period so that the counter is cleared. PALE is also raised high for the duration of the load. Next, MASN is asserted and the first word is accessed. After each DTACKINN, the MAC will dis-assert MASN for a 1-1/2 clock period and reissue MASN as many times as specified in the MFR register. As the high going edge of MASN will increment the counter, the correct word address within the block will be produced. At the end of the block load, the MAC will lower both PALE and LACLR so that the correct low order address is fed to the cache before asserting DTACKOUTN to the processor.

Note that if the SCC68172 BUSCON is used in the system and sequential accesses are necessary, then LACLR can be used to set an external 74F74 to indicate that a sequential access is being made. PALE going low at the end of the sequence will reset the flip-flop.

Write Through

When a write is accomplished, the information is written to memory. Information is also written to the cache only when a sub-cache is already allocated for this data and the relevant block presence bit is marked as valid.

Read/Write Overlap

On a write, the BMAC will strobe the address and data into external latches and start a

Advance Information

Table 3. BMAC BIT REGISTER MAP[1]

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
PDR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	pg size	
SSR	*	*	*	*	*	*	*	*	*	*	*	*	segment size			
SCR	*	*	*	*	*	*	*	*	*	*	*	*	cache organize			
ECR	T	S	E	R	W	*	*	*	P/D	EC4	EC3	EC2	EC1	EC0	RN1	RN0
TR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
PIR	*	*	*	*	*	*	*	*	*	*	*	*	processor no.			
MMP	*	*	*	*	*	*	*	*	*	*	*	*	*	*	enb1	cach
MFR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	count no.	
TMR	*	*	*	*	*	*	*	*	*	*	B5	B4	B3	B2	B1	B0
DBR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	nxt slot	
DESCR0	V	S	E	R	W	NC	K	C	SP	O	*	*	*	*	*	*
(Null)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(C = 0)	P	PS	PE	PR	PW	NC	D	U	LI3	LI2	LI1	LI0	*	*	*	*
DESCR1 (C = 1)	*	*	*	*	*	*	D	U	LI3	LI2	LI1	LI0	*	*	*	*
DESCR2 (PA)	*	*	23	22	21	20	19	18	17	16	15	14	13	12	11	10
(Null)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
(Null)	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR0U	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR0L	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR1U	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR1L	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR2U	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR2L	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR3U	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STPR3L	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RATR	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
RDR	*	region 2 top msbs					region 1 top msbs						region 0 top msbs			
STLR0	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STLR1	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STLR2	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
STLR3	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MCAPRU	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
MCAPRL	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

NOTE:
* Indicates null bits. All null bits will be read as zero.

memory write cycle. Before doing so, it will check if it has the required descriptor; if not, it will assert BERR. If it is not already set, the BMAC will also set the dirty bit in the page translation look-aside buffer on the BMAC and assert BERR to the processor. Then it will allow the processor to proceed by asserting the DTACK signal while the memory transfer is taking place. The processor will proceed unless it requests another memory transfer (either a cache read miss or a write) before the first write is complete.

Size of the Cache

The size of the cache is not infinitely variable, but can be expanded in both width and depth. The width can be expanded in units of 16 bits. The block can be configured to be 1, 2, 4, or 8 units wide, i.e., 16, 32, 64 or 128 bits wide. This width, the block size, is independent of the width of the data path to the processor. The number of memory accesses needed to fill a block is given by the memory fetch register.

The length of the cache can also vary and, again, the lengths to which it can be extended are limited. These limits are 512, 1024, or 2048 blocks deep. In terms of bytes the

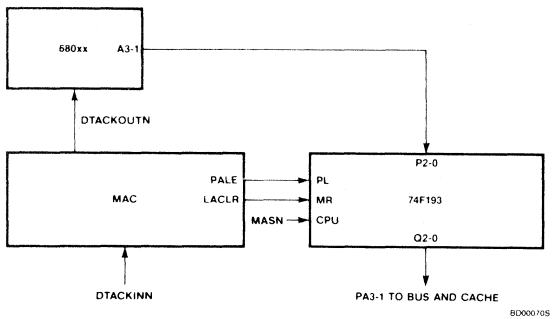


Figure 5. Cache Memory Block Transfer Circuitry

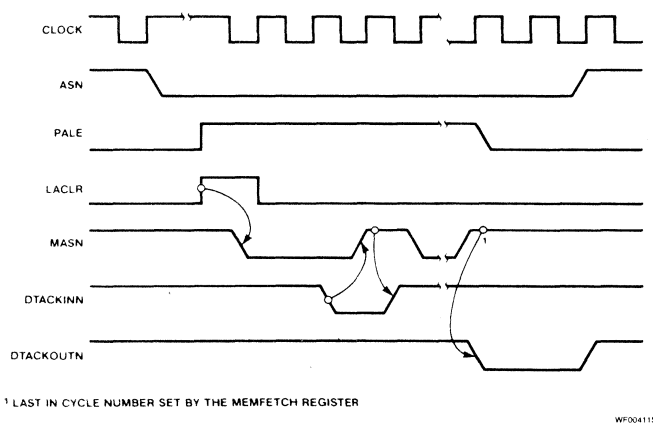


Figure 6. Memory Fetch Cycle Timing

cache can then vary from a minimum of 1K bytes (512 blocks of 2 bytes) to 32K bytes (2K blocks of 16 bytes).

Although the actual number is application dependent, in general, a cache of 1024 blocks of four words, i.e., 1Kx64, will result in the processor seeing a cache access time for at least 95% of the memory accesses. For example, if a cache hit or an overlappped write occur fast enough not to cause any wait states in the processor and a memory access results in four wait states, then the average number of wait states per memory access is 0.2. These figures are realistic for a 16MHz 68010/20 coupled to a slow, 250ns access memory.

Memory Mapping Unit

The memory mapping unit (MMU) performs paging control between three levels of memory. These three levels are:

1. Backing store (Disk)
2. System memory (RAM)
3. Local memory (RAM)

Local memory is not always present. In a multiprocessor situation, it may be desirable in order to reduce bus traffic, to attach a local memory to each or some of the processors. If so, the BMAC is capable of providing signals to the processor if the segment is marked as belonging to local memory. This page fault trap will be generated if the accessed page is not present in the local memory. Thus the BMAC cannot only control paging between the system memory and the backing store, but also between local memories and the system memory or backing store.

The memory mapping unit section of the BMAC contains descriptors for 32 pages which are loaded on demand by the BMAC. As in the cache, each page is identified only by its logical address. In addition, the MMU contains four register pairs. These consist of a segment table pointer register and a segment table length register. Each of these pairs define the current context of their respective regions. The region definition register is used to define region boundaries. The page descriptors are in a fully associative TLB on the BMAC. Each descriptor consists of the following fields:

1. An 18-bit physical address.
2. Four permission bits
   - Supervisor permission needed (S)
   - Access permission bits:
   - Read access (R)
   - Write access (W)
   - Execute access (E)
3. Page management bits
   - Non-cacheable (NC)
   - Local (L)
4. A dirty bit (D), i.e., the page has been written into.

The processor must write the descriptor to be used into the descriptor registers. This is done by loading one word of the relevant segment table into register DESCR0, and the two words of the relevant page table descriptor into register DESCR1 and register DESCR2, respectively. While the three registers DESCR0 - 2 are loaded automatically, the BMAC will check whether a page that is marked local is present in the required local memory. This is done only when the local bit in the segment table is set, by comparing the LI field in the page table with the contents of the PIR register. When the L bit is not set, or the LI field equals 1111, this check is disabled. The result of this check is stored in the ECR register. When the result is negative, i.e., the page is not in the required local memory, the BMAC will not load the page descriptor into its cache.

There is space for a cache of 32 descriptors on the BMAC. Each time a new descriptor is loaded, it is loaded into the next slot, modulo 31.

After the BMAC is reset, it will start putting descriptors in the TLB from slot 0. The first 1, 2, or 3 slots should be used for the descriptors which contain the system stack, BERR handling code, and data. These 'fixed' descriptors need to be resident before the virtual address system is enabled. It should be noted that regions containing these 'fixed' descriptors must not be flushed. After the 'fixed' descriptors have been loaded, the 2-bit DBR register must be loaded. This register indicates which descriptor slot is used after descriptor slot 31 is loaded, i.e., it ensures the 'fixed' descriptors are not overwritten. Thus, a

Advance Information

DBR register with a value of 2 would cause the next descriptor pointer to be updated to 2 after 31.

Translation Look-Aside Buffer Miss
If a word is read and it is not in the TLB, then the MMU section is accessed to find its physical address. If the relevant page descriptor is not in the MMU then, the BMAC will assert BERR.

Handling of the Dirty Bit
If any permitted write is made to a page and the dirty bit in the page descriptor in the TLB is not set, then it is set in the TLB and a BERR is generated.

Handling of Illegal Accesses to Memory
If an access is made to a memory location for which the required permission bits are not set, then the BERR is set. The reason is stored in the BMAC, in the error code register. If an access to a non-resident page is made, the same procedure will be followed. This will be done whether the page at fault was the desired page, or a page containing either the segment table or the page table.

Error Reporting
When an error or a page fault is detected by the BMAC during its operation, an error-report containing a description of the problem is stored in the ECR by the BMAC. The BMAC then generates a BERR to the processor.

THE BMAC's PLACE IN THE SYSTEM

It is possible to have one or more BMACs in the system. A system with one processor with an attached BMAC is shown in figure 7. The BMAC handles all the processor requests to memory and completely controls the cache. This is a fairly straight forward solution, as all the DMA requests use real addresses. While this is fairly straight forward for the hardware, the software has to arrange the blocks that have to be DMA'ed into memory to fit into consecutive real addresses, unless the DMA units can handle a scattered real memory.

Virtual I/O Addresses

This system is shown in figure 8. It is much easier for the software to handle, but the address bus onto which the DMA peripherals hang is the bus between the processor and the BMAC (note that all relevant I/O descriptors must be in the BMAC before needed). This is fine for a system with limited modularity, but it is difficult for a system which needs the traditional modularity, e.g., on the card level of a minicomputer. If that type of system is needed, then the configuration given in figure 9 can be used, i.e., with a separate BMAC dedicated to I/O. Here all descriptors are loaded before the start of an I/O action.

I/O units using a BMAC for address translation, must drive the FC3 signal low. The BMAC will then skip the check on access violations. FC2 – 0 can be used to identify each of up to seven I/O units connected to the MAC (note that FC2 – 0 = 000 is reserved and may not be used for I/O identification).

Multiple MACs Attached to Multiple Processors

This can be accomplished in two ways depending on the system needs. A typical system without local memory is shown in figure 10. This sort of system will allow up to four 68010s to work without seriously disturbing one another. If they work in a MIMD (multi-instruction stream, multi-data stream) mode, i.e., each working on its own process but only one running each section of the operating system at a time. Then typically four 68010s should give the performance of about 3.2 68010s, each running in their own system. This assumes that the processors are memory bound and not I/O bound. Note that although figure 10 shows processors each with their own MAC, these processors could just as easily be I/O units with DMA properties.

Systems with Local Memory

A typical system is shown in figure 11. Systems like this can be used if there is a way to partition the application. This can be done in many ways depending on the complexity of the system and the software. Some applications can be partitioned statically, e.g., a process control system where the tasks can be pre-assigned to a processor. In this case, the local memory can be used for code and process specific data, e.g., stacks.

Of course, applications that are reassigned dynamically can also be made using this system arrangement and the BMAC provides facilities to page in and out of local memories. When the page descriptor associated with a local page or segment (i.e., L bit is set) is loaded into the BMAC, an error status is stored in the ECR register if the LI field does not match the BMAC's PIR register contents. This check is overridden if the LI field is 1111.

The BMAC allows up to fifteen 680xxs, each having its own local memory to cooperate together. In particular, with the requisite system software, it is possible to write application programs that could run unchanged in either a single processor system or a multiprocessor with or without local memory.

The Use of Multiple BMACs Attached To One Processor

In some special cases, there may be a need to attach more than one BMAC to a processor. Normally, this should not be necessary as the cache hit rate should be high enough (greater than 90%) and the TLB hit rate should be better than 99.5%. However, if this is not enough or if a fast response is wanted for interrupts then multiple BMACs can be used. The BMAC will keep all of its output lines in a high impedance state when ASN is high. Therefore, multiple BMACs can be used by gating ASN with another address line(s). Note this address line should not be part of the offset within a page as that will reduce the effect of adding more than one BMAC.

If multiple BMACs are attached to one processor, then all the BMACs have to have identical region and STPR definitions.

PROGRAMMING THE BMAC

On power-up all the registers on the BMAC are reset to zero. This implies that there are no caches attached and all addresses are real, i.e., the BMAC does no translation whatsoever. The BMAC will also be reset by the RESET signal.

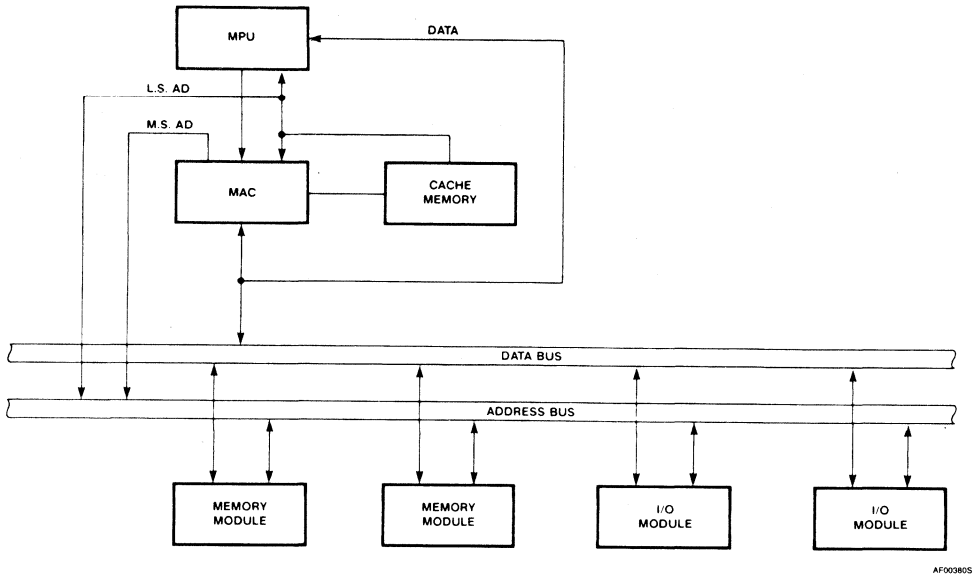


Figure 7. Processor With BMAC, Real I/O

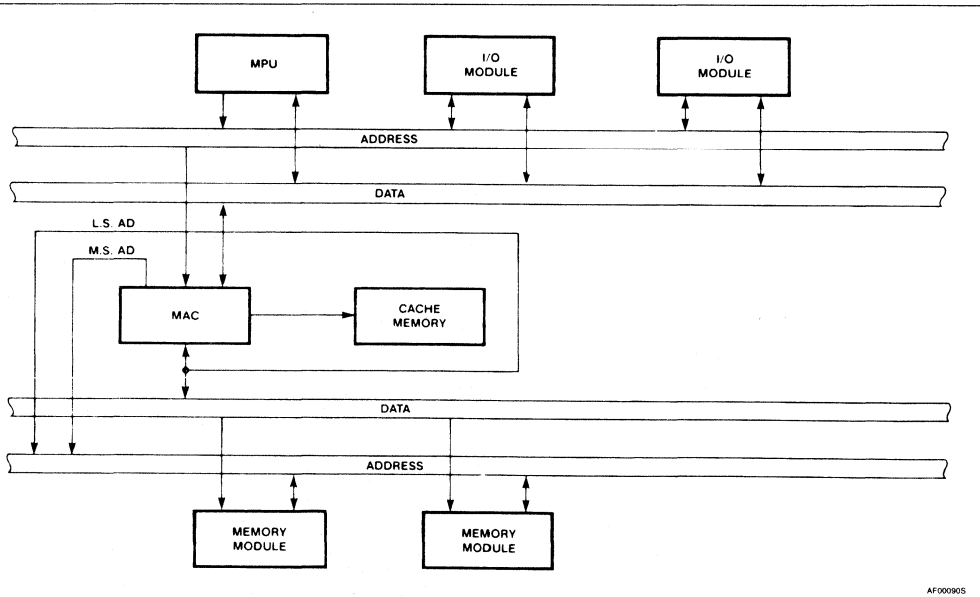


Figure 8. Virtual I/O

Advance Information

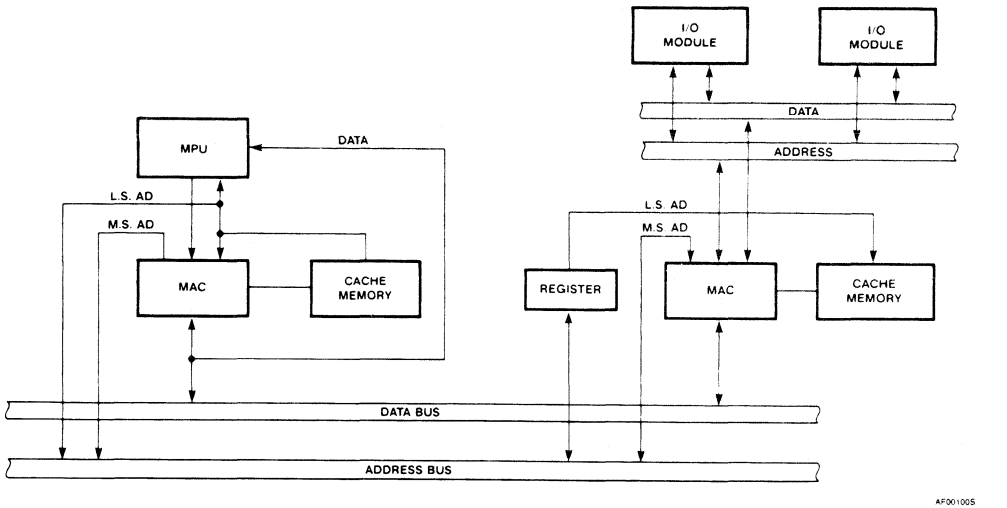


Figure 9. MAC Dedicated to I/O

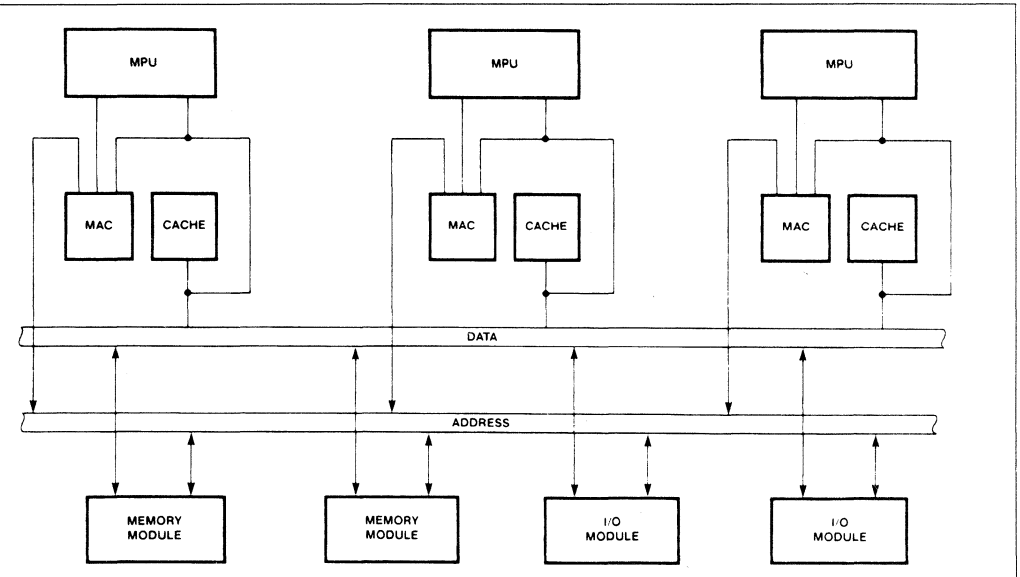


Figure 10. Multiple Processors, No Local Memory

Initializing the System

In order to remain compatible with the MAC system, the following registers have to be set.

1. The page definition register
2. The size of segment register
3. The region definition register
4. The segment table pointer registers
5. The segment table length registers
6. The BMAC communication area pointer register
7. The size of cache register
8. The memory fetch register
9. The BMAC mode register

The order of setting is irrelevant, except that as the BMAC mode register activates the system it should be set last. In addition the 'fixed' descriptors and DBR register have to be loaded on the BMAC.

Programming on a Context Change

As the BMAC needs only to be given a process context, it does not have to be reloaded on a process change. The only actions necessary are to invalidate the old context. This is done by loading the lower portion of the relevant region pointer register STPRnL. This action only invalidates the cache and TLB so as not to confuse the old context with the new context.

Programming for Change of System State

This is similar to the process change. The only difference is the setting of the STPR registers. Care should be taken when this is done. In particular, if there are multiple units accessing memory, i.e., processors or I/O units, they should be in a known state before this is done and should be suitably informed when the system state has changed. In other words, it is assumed that the system is normally only the kernel of the operating system and is normally fixed in memory.

This possibility for reconfiguration is only provided for safety and to allow the system to be reconfigured on a module failure.

Programming for I/O

Two philosophies are possible:

1. Real I/O
2. Virtual I/O

In real I/O, all I/O is done in real address space and the BMAC is not involved with the I/O operation. In virtual I/O, system I/O is mapped through one or more BMACs. If BMACs are shared between DMA control units then, seven function codes 0xxx should be shared, one to each of up to seven peripherals. Note that function code 0000 is reserved for the processor and cannot be used for I/O devices. Also note that FC3 is low for I/O transfers. The required descriptors should be in the BMAC. If the associated peripheral controller does not contain a deep FIFO, then required pages and tables should be locked in memory.

If required descriptors are not initially in the BMAC, then care must be taken to ensure their presence before they are needed.

TIMING

Memory accesses initiated by the BMAC will use the same timing as the 68000, i.e., three cycles for read and four for write. The internal timing of the BMAC will be such that when a series of reads have to be done, assuming zero wait states from the memory, then a read will be issued every four clock cycles. Some exceptions to this requirement may be made.

The CWEN signal will have a positive set-up and hold time with respect to CCEN.

ADDRESS FORMATS AND MEMORY ORGANIZATION SUMMARY

Table 4 summarizes the address and data path widths for the BMAC.

The logical address is composed of three fields: segment number, page number and offset within the page. The sizes of these

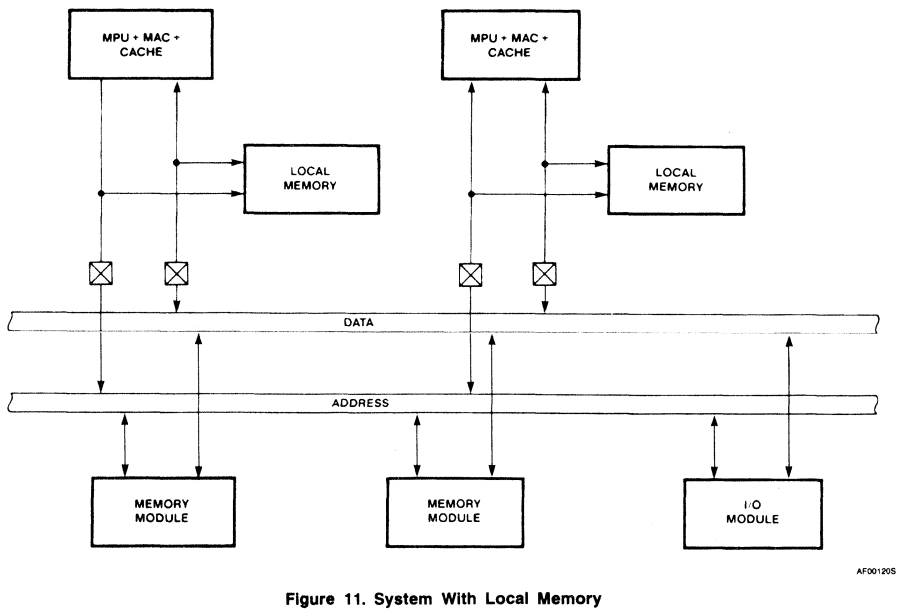


Figure 11. System With Local Memory

Advance Information

fields are programmable in the BMAC via the page definition register. The possible address compositions are shown in table 5.

BMAC BERR HANDLER PROCESSING

Refer to figure 12 which covers all of the possible BERR handling features. The BERR handler routine first saves any required 68010 registers. The contents of the error code register are then read. When bit 15 = 0, an access/permission/true BERR has occurred. Processing then branches to entry E1, where bits 2 – 6 of the ECR are checked for the type of violation. Acces s violations are further identified via bits 11 – 14. A jump table contains the entry points for the various violations. These violations are listed in table 6.

The main processing of the BMAC BERR handler occurs when ECR bit 15=1, indicating a descriptor fault. The routine then accesses the 68010 exception stack to obtain three words; the fault address (high/low) and the special status word (see table 7). Note that the routine may have to move this information from the stack in three 16-bit word moves, foregoing the more efficient long-word moves to avoid possible addressing (boundary) problems. The fault address words provide the virtual address of the BERR. These words must be saved in a 68010 data register. This data represents the segment number, page

Bit 8 of the special status word, figure 13, indicates if a read/write was being performed when the BERR occurred. Bits 0 – 1 of the ECR contain the region number where the error occurred. Applying the region number to the region definition register provides the segment table pointer for that region. Next, the routine will shift out the segment number from the virtual address.

Table 4. ADDRESS AND DATA PATH WIDTH

VERSION	NUMBER OF BITS		
	Logical ADDR	Physical ADDR	Data
84-PIN	24	24*	16

*If no local memories are used, the physical address may consist of 25 bits.

24-BIT LOGICAL ADDRESS

23	13	12	10	9	0
S/P -	S/P	P/0 - - - -	P/0	0 - - - - - - -	0

Table 5. MEMORY ORGANIZATION

SEGMENT SIZE (BYTES)	NUMBER OF SEGMENTS	MAX NUMBER OF PAGES PER SEGMENT			
		Page Size 1K Bytes	Page Size 2K Bytes	Page Size 4K Bytes	Page Size 8K Bytes
16M	1	16K	8K	4K	2K
8M	2	8K	4K	2K	1K
4M	4	4K	2K	1K	512
2M	8	2K	1K	512	256
1M	16	1K	512	256	128
512K	32	512	256	128	64
256K	64	256	128	64	32
128K	128	128	64	32	16
64K	256	64	32	16	8
32K	512	32	16	8	4
16K	1K	16	8	4	2
8K	2K	8	4	2	1

Table 6. BMAC BERR HANDLER SUBROUTINES FOR VIOLATIONS (E1 ENTRY POINTS)

BITS 2–6	BITS 11–14	SUBROUTINE ENTRY POINT
00000	NA	No error
00001	NA	Page not in required local memory
00010	 0001 0010 0100 1000	Access permission violation: Supervisor permission Execute access Read access Write access
01000	NA	Illegal BMAC access
01111	NA	BERR on local bus

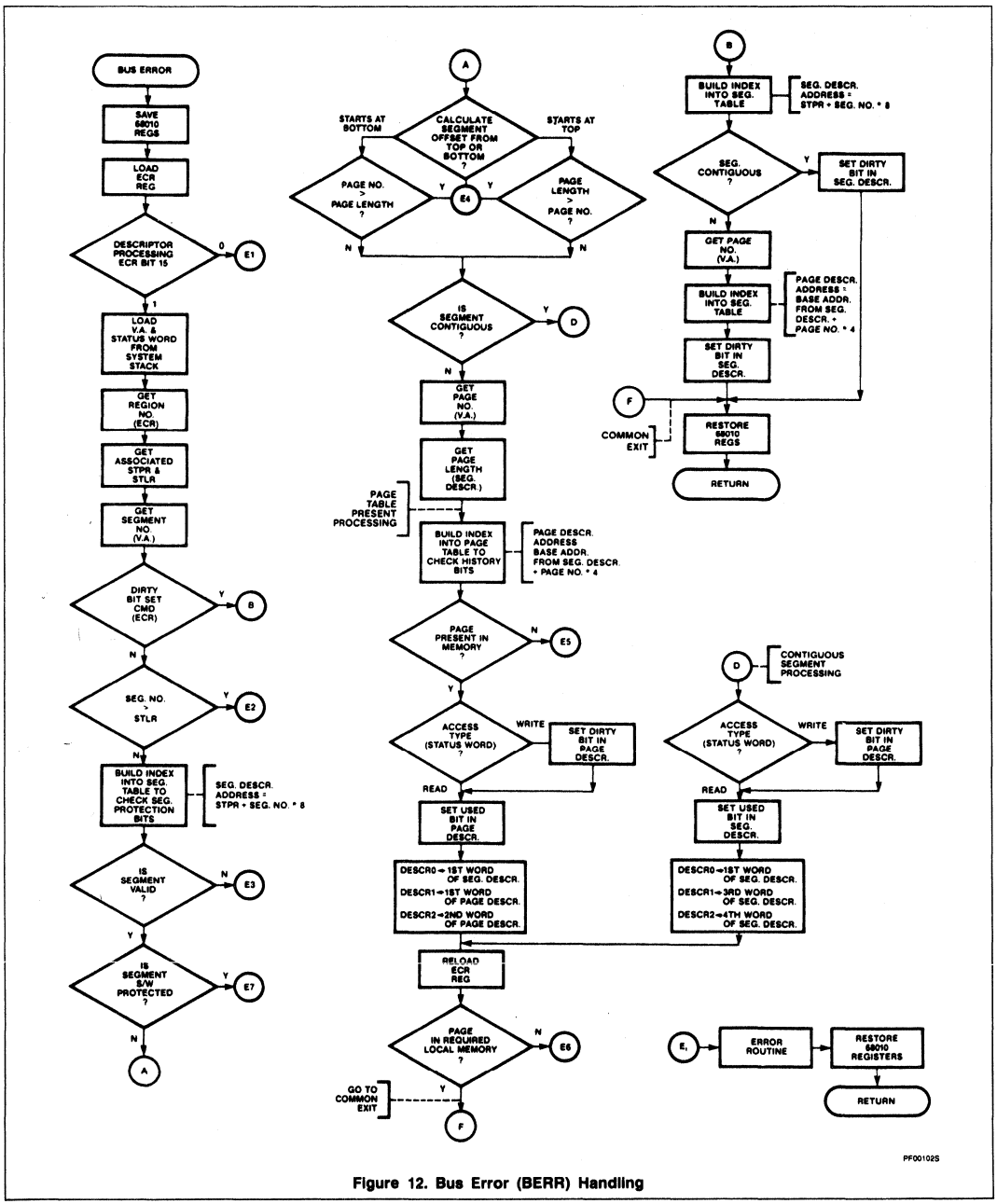


Figure 12. Bus Error (BERR) Handling

Advance Information

Bit 7 of the ECR now determines the major processing decision of the routine. If bit 7 = 1, a page descriptor miss has occurred. A series of segment and page checks need to be made, and then the three BMAC descriptor registers will be loaded by the routine. When bit 7 = 0, the dirty bit has not been set on a write. The series of checking and descriptor loading will be bypassed, as they already have been performed. Table 8 gives the possible error processing done by the routine. Both branches refer to the segment descriptors and the page descriptors.

When bit 7 = 1, the segment number (which has been shifted out of the virtual address by the routine) is compared to the number of segments currently valid (segment table length register). If the segment number is greater than STLR, a 'segment no. out-of-range' error is in effect, and the routine will branch to the E2 error entry point (see table 6). Otherwise, the segment descriptor is accessed to test the various protection bits. The segment descriptor address will be built by the routine as follows: STPR + segment number * 8.

If the segment is invalid (bit 31 = 1), processing branches to error entry point E3. If the software protection (bit 23) is set, the routine will branch to error entry point E4. Note that the SP bit is not used by the BMAC; it is an optional feature for use by the software.

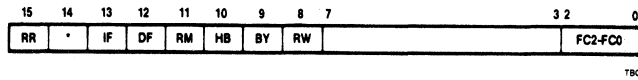
The offset (bit 22) bit denotes whether the pages of a segment start at the top (higher address, O = 1) or bottom (lower address, O not = 1) of the segment's virtual address space. The routine will use the page number (shifted out of the virtual address of the exception stack) and the page length (from the segment descriptor words). If O = 1, the routine will check for the page length being greater than the page number. If O is not = 1, the error checked will be for the page number being greater than the page length. Both error conditions will force a branch to error entry point E4. Note that offset error checking is optional for contiguous segments.

Next, the routine will test the contiguous bit (bit 24) in the segment descriptor. If C = 1, the segment is contiguous, i.e., has no page table. The routine will then skip over the page checking code to go on to loading the segment data into the descriptor registers. If C is not = 1, then the routine will perform page processing as follows. It will shift out the page number from the virtual address and get the page length from the segment descriptor. Then, the routine will build the page descriptor address thus: base address from segment descriptor + page number * 4.

Next the page present bit (bit 31) will be tested. If P = 0, the page is not present in memory, and the routine will branch to error entry point E5 to process the error. Otherwise, the routine will proceed with remaining page processing. The access type (from special status word) will then be tested. If a write, the dirty bit and the used bit in the page descriptor will be set. If the BERR interrupted

Table 7. EXCEPTION STACK ORDER FOR 68010 (BUS AND ADDRESS ERROR)

15	0
Program counter (high)	
Program counter (low)	
1000	Vector offset
Special status word	
Fault address (high)	
Fault address (low)	
Unused, reserved	
Data output buffer	
Unused, reserved	
Data input buffer	
Unused, reserved	
Instruction input buffer	
Internal information	



RR – Re-run flag; 0 = processor re-run (default), 1 = software re-run.
IF – Instruction fetch to the instruction input buffer.
DF – Data fetch to the data input buffer.
RM – Read-modify-write cycle.
HB – High byte transfer from the data output buffer or to the data input buffer.
BY – Byte transfer flag; HB selects the high or low byte of the transfer register. If BY is clear, the transfer is a word.
RW – Read/write flag; 0 = write, 1 = read.
FC – The function code used during the faulted access.
* These bits are reserved for future use and will be zero when written by the 68010.

Figure 13. Special Status Word Format

Table 8. ERROR ENTRY POINTS

ENTRY	SUBROUTINE ENTRY
E1	Access/other errors - 'good' violations (table 5)
E2	Segment number out of range i.e. greater than number of segments currently valid
E3	Segment not valid
E4	Page number out of range i.e., is in invalid address space of the segment
E5	Page is not present in memory
E6	Page is not in required local memory
E7	Segment descriptor error - S/W protection bit set

a read access, only the used bit will be set by the routine. Next, the routine will load the three descriptor registers, DESCR0–DESCR2, as follows: DESCR0 = first word of segment descriptor, DESCR1 = first word of page descriptor, and DESCR2 = second word of page descriptor. Two important rules must be followed here:

1. The registers must be loaded in the order of DESCR0, DESCR1, DESCR2.
2. The registers must be loaded atomically. Thus, the 68010 MOVEM instruction must be used.

The routine will then reload the contents of the ECR. This is required as loading DESCR0-DESCR2 causes the BMAC to check whether a page that is marked local is present in memory and to set bits 2 – 6 accordingly (bits 2 – 6 = 00001). If the bit has been set, the routine will branch to error entry point E6. Else, the routine will restore its 68010 registers and return.

In the case where the segment is contiguous (C = 1), the routine will set the dirty and used bits in the segment descriptor for a write access; the used bit will be set if the BERR occurred during a read access (access type being obtained from the special status word). The routine will then perform activities similar to those taken for loading page descriptors, except for the following differences. DESCR0 will be loaded with the first word of the segment descriptor, DESCR1 will be loaded with the third word of the segment descriptor, and DESCR2 will be loaded with the fourth word of the segment descriptor. The remaining processing is identical to that for paged segments.

The remaining paragraphs describe actions that will be performed when a dirty bit is not set on a write (ECR bit 7 = 0). The routine will build the segment descriptor address thus: STPR + Segment number * 8. Next, if the segment is contiguous (C = 1, bit 24 of the segment descriptor), the routine will set the dirty bit (bit 25) in the segment descriptor, restore the 68010 registers and return. If the segment has page tables, the page number will be shifted out of the virtual address and will be used to build the page descriptor address, thus: base address from segment descriptor + page number * 4. The routine will then set the dirty bit (bit 25) in the page descriptor, restore the 68010 registers and return.

Data communication for the SC68000 family

DUAL UNIVERSAL SERIAL COMMUNICATIONS CONTROLLER

Advance Information

Originally published by Signetics April 1984

DESCRIPTION

The SCN68562 DUSCC is a single chip MOS-LSI communications device that provides two independent, multi-protocol, full duplex receiver/transmitter channels in a single package. It supports bit oriented and character oriented (byte count and byte control) synchronous data link controls as well as asynchronous protocols. The SCN68562 interfaces to the 68000 MPU via asynchronous bus control signals and is capable of program polled, interrupt driven, or DMA data transfers.

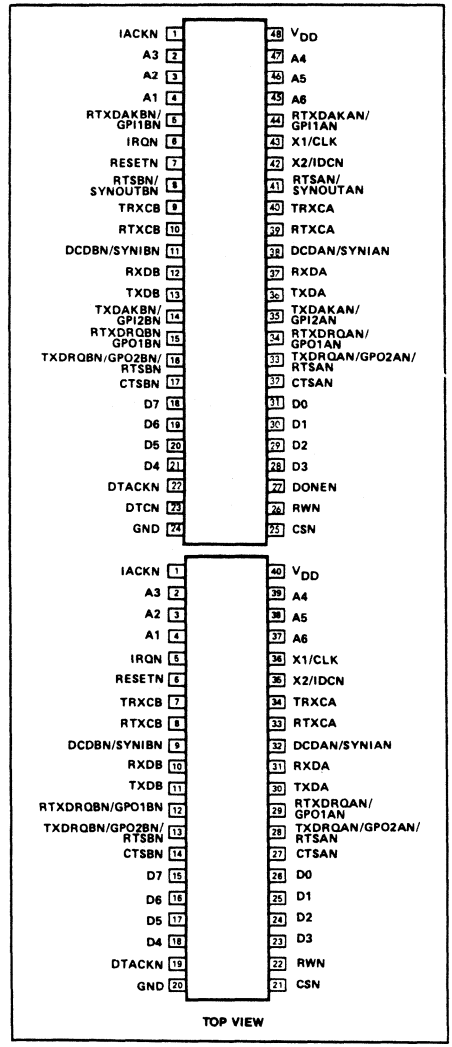
The operating mode and data format of each channel can be programmed independently. Each channel consists of a receiver, a transmitter, a 16-bit multi-function counter/timer, a digital phase locked loop (DPLL), a parity/CRC generator and checker, and associated control circuits. The two channels share a common bit rate generator (BRG), operating directly from a crystal or an external clock, which provides sixteen common bit rates simultaneously. The operating rate for the receiver and transmitter of each channel can be independently selected from the BRG, the DPLL, the counter/timer, or from an external 1x or 16x clock, making the DUSCC well suited for dual speed channel applications.

The transmitter and receiver each contain a four-deep FIFO with appended transmitter command and receiver status bits and a shift register. This permits reading and writing of up to four characters at a time, minimizing the potential of receiver overrun or transmitter underrun, and reducing interrupt or DMA overhead. In addition, a flow control capability is provided to disable a remote transmitter when the FIFO of the local receiving device is full.

Two modem control inputs (DCD and CTS) and three modem control outputs (RTS and two general purpose) are provided. Because the modem control inputs and outputs are general purpose in nature, optionally they can be programmed for other functions.

The DUSCC is available in a 40-pin and 48-pin DIP, and a 44-pin PLCC.

PIN CONFIGURATION



ORDERING CODE

PACKAGES	$V_{CC} = 5V \pm 5\%$, $T_A = 0°$ to 70°C		
	40-Pin	44-Pin	48-Pin
Ceramic DIP	SCN68562Cx140		SCN68562Cx148
Plastic DIP	SCN68562CxN40		SCN68562CxN48
Plastic LCC		SCN68562CxPLCC	

PLCC TO BE PROVIDED

Advance Information

FEATURES

General Features
- o Dual full-duplex synchronous/asynchronous receiver and transmitter
- o Multi-protocol operation
  - BOP: HDLC/ADCCP, SDLC, X.25 or X.75 link level, etc.
  - COP: BISYNC, DDCMP, X.21
  - ASYNC: 5-8 bits plus optional parity
- o Four character receiver and transmitter FIFOs
- o Programmable bit rate for each receiver and transmitter selectable from:
  - 16 fixed rates: 50 to 38.4K baud
  - One user defined rate derived from programmable counter/timer
  - External 1x or 16x clock
  - Digital phase locked loop
- o 0 to 4MHz data rate
- o Parity and FCS (frame check sequence) LRC or CRC generation and checking
- o Programmable data encoding/decoding: NRZ, NRZI, FM0, FM1, Manchester
- o Programmable channel mode: full/half duplex, auto-echo, or local loopback
- o Programmable data transfer mode: polled, interrupt, DMA, wait
- o DMA interface
  - Compatible with Signetics 68430 Direct Memory Access Interface (DMAI) and other DMA controllers
  - Half or full duplex operation
  - Single or dual address data transfers
  - Automatic frame termination on terminal count or DMA DONE
- o Interrupt capabilities
  - Daisy chain option
  - Vector output (fixed or modified by status)
  - Programmable internal priorities
  - 68K compatible
- o Multi-function programmable 16-bit counter/timer
  - Bit rate generator
  - Event counter
  - Count received or transmitted characters
  - Delay generator
  - Automatic bit length measurement
- o Modem controls
  - RTS, CTS, DCD, and up to four general purpose I/O pins per channel
  - CTS and DCD programmable auto-enables for Tx and Rx
  - Programmable interrupt on change of CTS or DCD
- o On-chip oscillator for crystal
- o TTL compatible
- o Single +5V power supply
- o 40/48-pin DIP

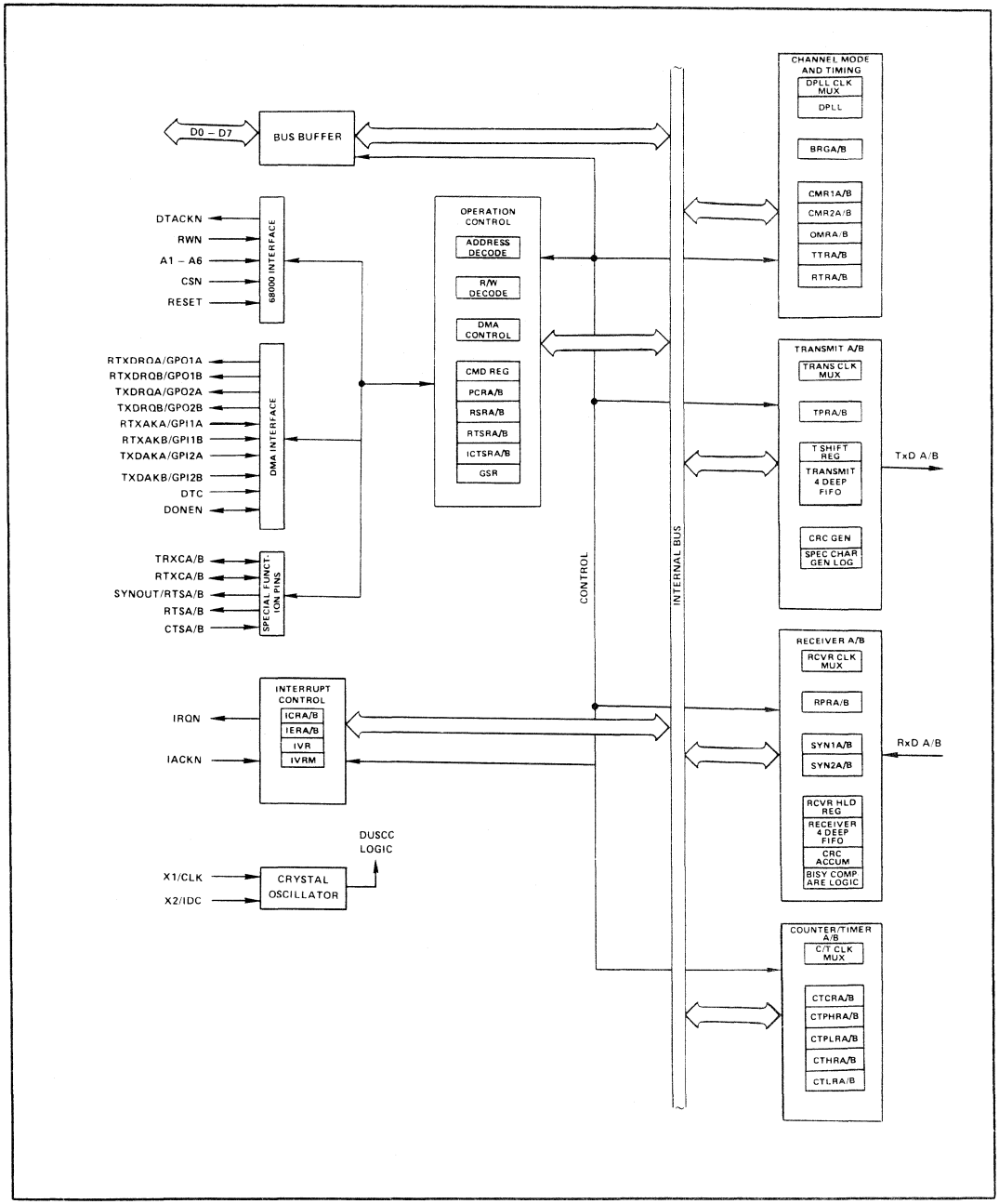
Asynchronous Mode Features
- o Character length: 5 to 8 bits
- o Odd or even parity, no parity, or force parity
- o Up to two stop bits programmable in 1/16 bit increments
- o 1x or 16x Rx and Tx clock factors
- o Parity, overrun, and framing error detection
- o False start bit detection
- o Start bit search 1/2 bit time after framing error detection
- o Break generation with handshake for counting break characters
- o Detection of start and end of received break
- o Character compare with optional interrupt on match

Character Oriented Protocol Features
- o Character length: 5 to 8 bits
- o Odd or even parity, no parity, or force parity
- o LRC or CRC generation and checking
- o Optional opening PAD transmission
- o One or two SYN characters
- o External sync capability
- o SYN detection and optional stripping
- o SYN or MARK linefill on underrun
- o Idle in MARK or SYNs
- o Parity, FCS, overrun, and underrun error detection
- o BISYNC Features
  - EBCDIC or ASCII text messages
  - SYN, DLE stripping
  - EOM (end of message) detection and transmission
  - Auto transparency mode switching
  - Auto hunt after receipt of EOM sequence
  - Control character sequence detection for both transparent and normal text

Bit Oriented Protocol Features
- o Character length: 5 to 8 bits
- o Detection and transmission of residual character: 0-7 bits
- o Automatic switch to programmed character length for I field
- o Zero insertion and deletion
- o Optional opening PAD transmission
- o Detection and generation of FLAG, ABORT, and IDLE bit patterns
- o Detection and generation of shared (single) FLAG between frames
- o Detection of overlapping (shared zero) FLAGs
- o ABORT, ABORT-FLAGs, or FCS-FLAGs line fill on underrun
- o Idle in MARK or FLAGs
- o Secondary address recognition including group and global address
- o Single or dual octet secondary address
- o Extended address and control fields
- o Short frame rejection for receiver
- o Detection and notification of received end of message
- o CRC generation and checking

Advance Information

BLOCK DIAGRAM



Advance Information

PIN DESCRIPTION

In this data sheet, signals are discussed using the terms 'active' and 'inactive' or 'asserted' or 'negated' independent of whether the signal is active in the high (logic 1) or low (logic 0) state. N at the end of a pin name signifies the signal associated with the pin is active low (see individual pin description for the definition of the active level of each signal.)

Pins which are provided for both channels are designated by either an underline (_) or by A/B after the name of the pin and the active low state indicator, N, if applicable.

A similar method is used for registers provided for both channels; these are designated by either an underline or by A/B after the name.

MNEMONIC	APPLICABLE 48	44	40	TYPE	DESCRIPTION
A1-A6	X	X	X	I	Address Lines: Active high. Address inputs which specify which of the internal registers is accessed for read/write operations.
D0-D7	X	X	X	I/O	Bidirectional Data Bus: Active high, three state. Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the DUSCC take place over this bus. The data bus is enabled when CSN is low, during interrupt acknowledge cycles and single-address DMA acknowledge cycles.
RWN	X	X	X	I	Read/Write: A high input indicates a read cycle and a low input indicates a write cycle when a cycle is initiated by assertion of the CSN input.
CSN	X	X	X	I	Chip Select. Active low input. When low, data transfers between the CPU and the DUSCC are enabled on D0-D7 as controlled by the R/WN and A1-A6 inputs. When CSN is high, the DUSCC is isolated from the data bus (except during interrupt acknowledge cycles and single-address DMA transfers) - D0-D7 are placed in the three state condition.
DTACKN	X	X	X	O	Data Transfer Acknowledge: Active low, three state. DTACKN is asserted on a write cycle to indicate that the data on the bus has been latched, and on a read cycle or interrupt acknowledge cycle to indicate valid data is on the bus. The signal is negated when completion of the cycle is indicated by negation of the CSN or IACKN input, and returns to the inactive (three state) a short period after it is negated. When negated, DTACKN becomes an open drain output and requires an external pull up resistor.
IRQN	X	X	X	O	Interrupt Request: Active low, open drain. This output is asserted upon occurrence of any enabled interrupting condition. The CPU can read the general status register to determine the interrupting condition(s), or can respond with an interrupt acknowledge cycle to cause the DUSCC to output an interrupt vector on the data bus.
IACKN	X	X	X	I	Interrupt Acknowledge: Active low. When IACKN is asserted, the DUSCC responds by placing the content of the interrupt vector register (modified or unmodified by status) on the data bus and asserting DTACKN. If no active interrupt is pending, DTACKN is not asserted.

Advance Information

MNEMONIC	APPLICABLE 48	44	40	TYPE	DESCRIPTION
X1/CLK	X	X	X	I	Crystal or External Clock: When using the crystal oscillator, the crystal is connected between pins X1 and X2. If a crystal is not used, an external clock is supplied at this input. This clock is used to drive the internal bit rate generator, as an optional input to the counter/timer or DPLL, and to provide other required clocking signals.
X2/IDCN	X	X	X	O	Crystal or Interrupt Daisy Chain: Active low. When a crystal is used as the timing source, the crystal is connected between pins X1 and X2. This pin can be programmed to provide an interrupt daisy chain output which propagates the IACKN signal to lower priority devices, if no active interrupt is pending.
RESETN	X	X	X	I	Master Reset: Active low. A low on this pin resets the transmitters and receivers and resets the registers shown in table 1.
RXDA, RXDB	X	X	X	I	Channel A (B) Receiver Serial Data Input: The least significant bit is received first. If external receiver clock is specified for the channel, the input is sampled on the rising edge of the clock.
TXDA, TXDB	X	X	X	O	Channel A (B) Transmitter Serial Data Output: The least significant bit is transmitted first. This output is held in the marking (high) condition when the transmitter is disabled or when the channel is operating in local loopback mode. If external transmitter clock is specified for the channel, the data is shifted on the falling edge of the clock.
RTXCA, RTXCB	X	X	X	I/O	Channel A (B) Receiver/Transmitter Clock. As an input, it can be programmed to supply the receiver, transmitter, counter/timer, or DPLL clock. As an output, can supply the counter/timer output, the transmitter shift clock (1X), or the receiver sampling clock (1X). The maximum external receiver/transmitter clock frequency is 4MHz.
TRXCA, TRXCB	X	X	X	I/O	Channel A (B) Transmitter/Receiver Clock: As an input, it can supply the receiver, transmitter, counter/timer, or DPLL clock. As an output, it can supply the counter/timer output, the DPLL output, the transmitter shift clock (1X), the receiver sampling clock (1X), the transmitter BRG clock (16X), or the receiver BRG clock (16X). The maximum external receiver/transmitter clock frequency is 4MHz.
CTSAN, CTSBN	X	X	X	I	Channel A (B) Clear-To-Send: Active low. The signal can be programmed to act as an enable for the transmitter. The DUSCC detects logic level transitions on this input and can be programmed to generate an interrupt when a transition occurs.

Advance Information

MNEMONIC	APPLICABLE 48	44	40	TYPE	DESCRIPTION
DCDA/BN, SYNIA/BN	X	X	X	I	Channel A (B) Data-Carrier-Detected or Extern Sync Input: Active low. The function of this pin is programmable. As a DCD input, it acts as an enable for the receiver or can be used as a general purpose input. For the DCD function, the DUSCC detects logic level transitions of this input and can be programmed to generate an interrupt when a transition occurs. As an external sync input, it is used in synchronous modes to obtain character synchronization without receipt of a SYN or FLAG character. This mode can be used in disc or tape controller applications or for the optional byte timing lead in X.21.
RTXDRQA/BN GPOIA/BN	X	X	X	O	Channel A (B) Receiver/Transmitter DMA Service Request or General Purpose Output: Active low. For half-duplex DMA operation, this output indicates to the DMA controller that one or more characters are available in the receiver FIFO (when the receiver is enabled) or that the transmit FIFO is not full (when the transmitter is enabled). For full-duplex DMA operation, this output indicates to the DMA controller that data is available in the receiver FIFO. In non-DMA mode, this pin is a general purpose output that can be asserted and negated under program control.
TXDRQA/BN GPO2A/BN RTSA/BN	X	X	X	O	Channel A (B) Transmitter DMA Service Request, General Purpose Output, or Request-to-Send: Active low. For full-duplex DMA operation, this output indicates to the DMA controller that the transmit FIFO is not full and can accept more data. When not in full-duplex DMA mode, this pin can be programmed as a general purpose or a Request-to-Send, which can be asserted and negated under program control (see Detailed Operation).
RTXDAKA/BN GPI1A/BN	X			I	Channel A (B) Receiver/Transmitter DMA Acknowledge or General Purpose Input: Active low. For half-duplex single address DMA operation, this input indicates to the DUSCC that the DMA controller has acquired the bus and that the requested bus cycle (read receiver FIFO or load transmitter FIFO) is beginning. For full-duplex single address DMA operation, this input indicates to the DUSCC that the DMA controller has acquired the bus and that the requested read receiver FIFO bus cycle is beginning. Because the state of this input can be read under program control, it can be used as a general purpose input when not in full duplex single address DMA mode.
TXDAKA/BN GPI2A/BN	X			I	Channel A (B) Transmitter DMA Acknowledge or General Purpose Input: Active low. When the channel is programmed for full-duplex single address DMA operation, this input is asserted to indicate to the DUSCC that the DMA controller has acquired the bus and that the requested load transmitter FIFO bus cycle is beginning. Because the state of this input can be read under program control, it can be used as a general purpose input when not in full duplex single address DMA mode.
DTCN	X			I	Device Transfer Complete: Active low. DTCN is asserted by the DMA controller to indicate that the requested data transfer is complete.
DONEN	X			I/O	Done: Active low, three state. See Detailed Operation for a descripiton of the function of this pin.

Advance Information

MNEMONIC	APPLICABLE 48	44	40	TYPE	DESCRIPTION
RTSA/BN SYNOUTA/BN	X			O	Channel A (B) Sync Detect or Request-to-Send: Active low. If programmed as a sync output, it is asserted whenever the specified sync character (COP or BISYNC modes) or a FLAG (BOP modes) is detected by the receiver. As a Request-to-Send modem control signal, it behaves as described previously for the TXDRQ_N/RTS_N pin. It performs the RTS function if the DUSCC is in full duplex single address DMA mode
V_{DD}	X	X	X	I	+5V ±5% power input.
GND	X	X	X	I	Signal and power ground input.

REGISTERS

The addressable registers of the DUSCC are shown in table 1. The following rules apply to all registers:

1. A read from a reserved location in the map results in a read from the 'null register'. The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle without a write being performed.
2. Unused bits of a defined register are read as zeros.
3. Bits that are unused in the chosen mode but are used in others are readable and writable but their contents are ignored in the chosen mode.
4. All registers are addressable as 8-bit quantities. To facilitate operation with the 68K MOVEP instruction, addresses are ordered such that certain sets of registers may also be accessed as words or long words.

The operation of the DUSCC is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The contents of certain control registers are initialized on RESET (set to zero). Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems, e.g., changing the channel mode at an inappropriate time may cause the reception or transmission of an incorrect character. In general, the contents of registers which control transmitter or receiver operation should be changed only when they are not enabled. The DUSCC registers can be separated into five groups to facilitate their usage:

1. Channel configuration and pin description registers
2. Transmitter and receiver parameter and timing registers
3. Counter/timer control and value registers
4. Interrupt control and status registers
5. Command register

This arrangement is used in the following description of the DUSCC registers.

MODE CONFIGURATION AND PIN DESCRIPTION REGISTERS

There are five registers in this group for each channel. The bit format for each of these registers is contained in table 2. The primary function of these registers is to define configuration of the channels and the function of the programmable pins.

Channel Mode Register 1 (CMR1A, CMR1B)

[7:6] Data Encoding

These bits select the data encoding for the received and transmitted data:

00 If the DPLL is set to NRZI mode (see DPLL commands), it selects positive logic (1 = high, 0 = low). If the DPLL is set to FM mode (see DPLL commands), Manchester (bi-phase level) encoding is selected.
01 NRZI. Non-return-to-zero inverted.
10 FM0. Bi-phase space.
11 FM1. Bi-phase mark.

[5] Extended Control (BOP)

0 No. A one-octet control field follows the address field.
1 Yes. A two-octet control field follows the address field.

[5] Parity (COP/Async), Code Select (BISYNC)

0 Odd parity if with parity is selected by [4:3] or a 0 in the parity bit position if force parity is selected by [4:3]. In BISYNC protocol mode, internal character comparisons are made using EBCDIC coding.

1 Even parity if with parity is selected by [4:3] or a 1 in the parity bit position if force parity is selected by [4:3]. In BISYNC protocol mode, internal character comparisons are made using 8-bit ASCII coding.

Advance Information

TABLE 1. DUSCC ADDRESS MAP

Address Bits[1] 6 5 4 3 2 1	Acronym	Affected by Register Name	Mode	Reset
c 0 0 0 0 0	CMR1	Channel Mode Register 1	R/W	Yes - 00
c 0 0 0 0 1	CMR2	Channel Mode Register 2	R/W	Yes - 00
c 0 0 0 1 0	S1R	SYN 1/Secondary Address 1 Register	R/W	No
c 0 0 0 1 1	S2R	SYN 2/Secondary Address 2 Register	R/W	No
c 0 0 1 0 0	TPR	Transmitter Parameter Register	R/W	Yes - 00
c 0 0 1 0 1	TTR	Transmitter Timing Register	R/W	No
c 0 0 1 1 0	RPR	Receiver Parameter Register	R/W	Yes - 00
c 0 0 1 1 1	RTR	Receiver Timing Register	R/W	No
c 0 1 0 0 0	CTPRH	Counter/Timer Preset Register High	R/W	No
c 0 1 0 0 1	CTPRL	Counter/Timer Preset Register Low	R/W	No
c 0 1 0 1 0	CTCR	Counter/Timer Control Register	R/W	No
c 0 1 0 1 1	OMR	Output and Miscellaneous Register	R/W	Yes
c 0 1 1 0 0	CTH	Counter/Timer High	R	No
c 0 1 1 0 1	CTL	Counter/Timer Low	R	No
c 0 1 1 1 0	PCR	Pin Configuration Register	R/W	Yes
c 0 1 1 1 1	CCR	Channel Command Register	R/W	No
c 1 0 0 x x	TXFIFO	Transmitter FIFO	W	No
c 1 0 1 x x	RXFIFO	Receiver FIFO	R	No
c 1 1 0 0 0	RSR	Receiver Status Register	R/W[2]	Yes
c 1 1 0 0 1	TRSR	Transmitter and Receiver Status Register	R/W[2]	Yes
c 1 1 0 1 0	ICTSR	Input and Counter/Timer Status Register	R/W[2]	Yes
d 1 1 0 1 1	GSR	General Status Register	R/W[2]	Yes
c 1 1 1 0 0	IER	Interrupt Enable Register	R/W	Yes
c 1 1 1 0 1		Not used		
0 1 1 1 1 0	IVR	Interrupt Vector Register - Unmodified	R/W	Yes
1 1 1 1 1 0	IVRM	Interrupt Vector Register - Modified	R	Yes
0 1 1 1 1 1	ICR	Interrupt Control Register	R/W	Yes
1 1 1 1 1 1		Not used		

NOTES:
1. c = 0 for channel A, c = 1 for channel B.
   d = don't care - register may be accessed as either channel.
   x = don't care - FIFOs are addressable at any of four adjacent addresses to allow them to be addressed as byte/word/long word with the 68K MOVEP instruction.

2. A write to this register may perform a status resetting operation.

Advance Information

TABLE 2. CHANNEL CONFIGURATION/PIN DEFINITION REGISTERS BIT FORMATS

CHANNEL MODE REG 1

(CMR1A, CMR1B)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DATA ENCODING		EXTENDED CONTROL	ADDRESS MODE (BOP)		CHANNEL PROTOCOL MODE		
		BOP only	00 - 8 bit		000 - BOP Primary		
00 - NRZ/Manch		0 - no	01 - extended addr		001 - BOP Secondary		
01 - NRZI		1 - yes	10 - 16 bit		010 - Reserved		
10 - FM0			11 - 16 bit w/group		011 - Reserved		
11 - FM1		# PARITY	PARITY MODE		100 - COP dual SYN		
			00 - no parity		101 - COP dual SYN (BISYNC)		
		0 - even	01 - reserved		110 - COP single SYN		
		1 - odd	10 - with parity		111 - asynchronous		
			11 - force parity				

Note: In BISYNC protocol mode, 0 = EBCDIC, 1 = 8-bit ASCII coding.

CHANNEL MODE REG 2

(CMR2A, CMR2B)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CHANNEL CONNECTION		DATA TRANSFER INTERFACE			FRAME CHECK SEQUENCE SELECT		
		000 - half dup sing add DMA			000 - none		
00 - normal		001 - half dup dual add DMA			001 - reserved		
01 - auto echo		010 - full dup sing add DMA			010 - LRC8 preset 0s		
10 - local loop		011 - full dup dual add DMA			011 - LRC8 preset 1s		
11 - reserved		100 - wait on Rx only			100 - CRC 16 preset 0s		
		101 - wait on Tx only			101 - CRC 16 preset 1s		
		110 - wait on Rx or Tx			110 - CRC CCITT preset 0s		
		111 - polled or interrupt			111 - CRC CCITT preset 1s		

SYN1/SECONDARY ADDRESS REG 1

(S1RA, S1RB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYN - COMPARE CHARACTER COP - SYN1 (5-8 BITS) BOP - FIRST ADDRESS OCTET							

SYN2/SECONDARY ADDRESS REG 2

(S2RA, S2RB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYN - NOT USED COP - SYN2 (5-8 BITS) BOP - SECOND ADDRESS OCTET							

PIN CONFIGURATION REG

(PCRA, PCRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
X2/IDC *	GPO2/ RTS	SYNOUT/ RTS	RTXC PIN		TRXC PIN		
			00 - input		000 - input	100 - TXCLK 16x	
0 - X2	0 - GPO2	0- SYNOUT	01 - C/T		001 - XTAL/2	101 - RXCLK 16x	
1 - IDC	1 - RTS	1 - RTS	10 - TXCLK 1x		010 - DPLL	110 - TXCLK 1x	
			11 - RXCLK 1x		011 - C/T	111 - RXCLK 1x	

*PCRA only. Not used in PCRB.

Advance Information

[4:3] Address Mode (BOP)
This field controls whether a single octet or multiple octets follow the opening FLAG(s) for both the receiver and the transmitter. This field is activated by selection of BOP secondary mode through the channel protocol mode bits CMR1 [2:0] (see Detailed Operation).

00 Single address octet.
01 Extended address.
10 Dual address octet.
11 Dual address octet with group.

[4:3] Parity Mode (COP/Async)
This field selects the parity mode for both the receiver and the transmitter. A parity bit is added to the programmed character length if with parity or force parity is selected:

00 No parity. Required when BISYNC protocol mode is programmed.
01 Reserved.
10 With parity. Odd or even parity is selected by [5].
11 Force parity. The parity bit is forced to the state selected by [5].

[2:0] Channel Protocol Mode
This field selects the operational protocol and submode for both the receiver and transmitter:
000 BOP Primary. No address comparison is performed. For receive, all characters received after the opening FLAG(s) are transferred to the FIFO.
001 BOP Secondary. This mode actives the address modes selected by [4:3]. Except in the case of extended address ([4:3]=01), an address comparison is performed to determine if a frame should be received. Refer to Detailed Operation for details of the various addressing modes. If a valid comparison occurs, the receiver is activated and the address octets and all subsequent received characters of the frame are transferred to the receive FIFO.

010 Reserved.
011 Reserved.
100 COP Dual SYN. Character sync is achieved upon receipt of a bit sequence matching the contents of the appropriate bits of S1R and S2R (SYN1-SYN2), including parity bits if any.
101 COP Dual SYN (BISYNC). Character sync is achieved upon receipt of a bit sequence matching the contents of the appropriate bits of S1R and S2R (SYN1-SYN2), including parity bits if any. In this mode, special transmitter and receiver logic is activated. (see Detailed Operation).
110 COP Single SYN. Character sync is achieved upon receipt of a bit sequence matching the contents of the appropriate bits of S1R (SYN1), including parity bit if any.
111 Asynchronous. Start/stop format.

Channel Mode Register 2 (CMR2A, CMR2B)

[7:6] Channel Connection
This field selects the mode of operation of the channel. The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character.

00 Normal mode. The transmitter and receiver operate independently in either half or full-duplex, controlled by the respective enable commands.
01 Automatic echo mode. Automatically retransmits the received data with a two bit time delay. The following conditions are true while in automatic echo mode:
1. Received data is reclocked and retransmitted on the TXDA output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but th transmitter need not be enabled.
4. The TXRDY and underrun status bits are inactive.
5. The received parity and/or FCS are checked if required, but are not regenerated for transmission, i.e., transmitted parity and/or FCS are as received.
6. In async mode, character framing is checked, but the stop bits are retransmitted as received. A received break is echoed as received.
7. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

10 Local loopback mode. In this mode:
1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver if NRZI or NRZ encoding is used. For FM or Manchester encoding because the receiver clock is derived from the DPLL, the DPLL source clock must be maintained.
3. The TXD output is held high.
4. The RXD input is ignored.
5. The receiver and transmitter must be enabled.
6. CPU to transmitter and receiver communications continue normally.

11 Reserved.

Advance Information

[5:3] Data Transfer Interface
This field specifies the type of data transfer between the DUSCC's Rx and Tx FIFOs and the CPU. All interrupt and status functions operate normally regardless of the data transfer interface programmed. Refer to Detailed Operation for details of the various DMA transfer interfaces.

000 Half duplex single address DMA. Valid for 48-pin version only.
001 Half duplex dual address DMA.
010 Full duplex single address DMA. Valid for 48-pin version only.
011 Full duplex dual address DMA.
100 Wait on receive only. In this mode a read of a non-empty receive FIFO results in a normal bus cycle. However, if the receive FIFO of the channel is empty when a read Rx FIFO cycle is initiated, the DTACKN output remains negated until a character is received and loaded into the FIFO. DTACKN is then asserted and the cycle is completed normally.
101 Wait on transmit only. In this mode a write to a non-full transmit FIFO results in a normal bus cycle. However, if the transmit FIFO of the channel is full when a write Tx FIFO cycle is initiated, the DTACKN output remains negated until a FIFO position becomes available for the new character. DTACKN is then asserted and the cycle is completed normally.
110 Wait on transmit and receive. Same as above for both wait on receive and transnit operations.
111 Polled or interrupt. DMA and wait functions of the channel are not activated. Data transfers to the Rx and Tx FIFOs are via normal bus read and write cycles in response to polling of the status registers and/or interrupts.

[2:0] Frame Check Sequence Select
This field selects the optional frame check sequence (FCS) to be appended at the end of a transmitted frame. The selected FCS is transmitted as follows:

1. Following the transmission of a FIFO'ed character tagged with the 'send EOM' command.
2. If underrun control (TPR[7:6]) is programmed for TEOM, upon occurrence of an underrun.
3. If TEOM on zero count or done (TPR[4]) is asserted and the counter/timer is counting transmitted characters, after transmission of the character which causes the counter to reach zero count.
4. In DMA mode with TEOM on zero count or done (TPR[4]) set, after transmission of a character if DONEN is asserted when that character was loaded into the Tx FIFO by the DMA controller.

000 No frame check sequence.
001 Reserved
010 LRC8: Divisor = x^8+1, dividend preset to zeros. The Tx sends the calculated LRC noninverted. The Rx indicates an error if the computed LRC is not equal to 0.
011 LRC8: Divisor = x^8+1, dividend preset to ones. The Tx sends the calculated LRC noninverted. The Rx indicates an error if the computed LRC is not equal to 0.
100 CRC16: Divisor = $x^{16}+x^{15}+x^2+1$, dividend preset to zeros. The Tx sends the calculated CRC noninverted. The Rx indicates an error if the computed CRC is not equal to 0.
101 CRC16: Divisor = $x^{16}+x^{15}+x^2+1$, dividend preset to ones. The Tx sends the calculated CRC noninverted. The Rx indicates an error if the computed CRC is not equal to 0.
110 CRC-CCITT: Divisor = $x^{16}+x^{12}+x^5+1$, dividend preset to zeroes. The Tx sends the calculated CRC noninverted. The Rx indicates an error if the computed CRC is not equal to 0.
111 CRC-CCITT: Divisor = $x^{16}+x^{12}+x^5+1$, dividend preset to ones. The Tx sends the calculated CRC inverted. The Rx indicates an error if the computed CRC is not equal to H'F0B8'.

SYN1/Secondary Address 1 Register (S1RA, S1RB)

[7:0] Character Compare
In async mode this register holds a bit pattern which is compared with received characters. If a match occurs, the character compare status bit (RSR[7]) is set. Parity need not be included in the value placed in the register. However, a character received with parity error will not match. This field is ignored if in a break condition.

In COP modes, this register contains the 5 to 8 bit SYN1 bit pattern, right justified. In BOP secondary mode it contains the address used to compare the first received address octet. The register is not used in BOP primary mode or secondary modes where address comparisons are not made, such as when extended addressing is specified.

Advance Information

SYN2/Secondary Address 2 Register (S2RA, S2RB)

[7:0]
This register is not used in async, COP single SYN, BOP primary modes, BOP secondary modes with single address field, and BOP secondary modes where address comparisons are not made, such as when extended addressing is specified.

In COP dual SYN modes, it contains the 5 to 8 bit SYN2 bit pattern, right justified. In BOP secondary mode using two address octets, it contains the partial address used to compare the second received address octet.

Pin Configuration Register (PCRA, PCRB)

This register selects the functions for multi-purpose I/O pins.

[7] X2/IDC
This bit is defined only for PCRA. It is not used in PCRB.
0 The X2/IDCN pin is used as a crystal connection.
1 The X2/IDCN pin is the interrupt daisy chain output.

[6] GPO2/RTS
The function of this pin is programmable only when not operating in full duplex DMA mode.

0 The TXDRQ_N/GPO2_N/RTS_N pin is a general purpose output. It is low when OMR[2] is a 1 and high when OMR[2] is 0.
1 The pin is a request-to-send output (see Detailed Operation).

[5] SYNOUT/RTS (48-pin version only)
The logical state of the pin is controlled by OMR[0], when set the output is zero.
0 The SYNOUT_N/RTS_N pin is an active low output which is asserted for one bit time when a SYN pattern (COP modes) or FLAG (BOP modes) is received.
1 The pin is a request-to-send output (see Detailed Operation).

[4:3] RTXC
00 The pin is an input. It must be programmed for input when used as the input for the receiver or transmitter clock, the DPLL, or the C/T.
01 The pin is an output from the counter/timer. Refer to CTCRA/B description.
10 The pin is an output from the transmitter shift register clock.
11 The pin is an output from the receiver shift register clock.

[2:0] TRXC
000 The pin is an input. It must be programmed for input when used as the input for the receiver or transmitter clock, the DPLL, or the C/T.
001 The pin is an output from the crystal oscillator. (XTAL/2)
010 The pin is an output from the DPLL output clock.
011 The pin is an output from the counter/timer. Refer to CTCRA/B description.
100 The pin is an output from the selected transmitter BRG frequency divided by two.
101 The pin is an output from the selected receiver BRG frequency divided by two.
110 The pin is an output from the transmitter shift register clock.
111 The pin is an output from the receiver shift register clock.

TRANSMITTER AND RECEIVER PARAMETER AND TIMING REGISTERS

This set of five registers contains the information which controls the operation of the transmitter and receiver for each channel. Table 3 shows the bit map format each of these registers. The registers of this group are:

1. Transmitter parameter and timing registers (TPRA/B and TTRA/B)
2. Receiver parameter and timing registers (RPRA/B and RTRA/B)
3. Output and miscellaneous register (OMRA/B)

The first and second group of registers define the transmitter and receiver parameters and timing. Included in the receiver timing registers are the programming parameters for the DPLL. The last register of the group, OMR, contains additional transmitter and receiver information and controls the logical state of the output pins when they are not used as a part of the channel configuration.

Transmitter Parameter Register (TPRA, TPRB)

[7:6] Underrun Control
In BOP and COP modes, this field selects the transmitter response in the event of an underrun (i.e., the Tx FIFO is empty).
00 Normal end of message termination. In BOP, the transmitter sends the FCS (if selected by CMR2[2:0]) followed by a FLAG and then either MARKs or FLAGs, as specified by [5]. In COP, the transmitter sends the FCS (if selected by CMR2[2:0]) and then either MARKs or SYNs, as specified by [5].

Advance Information

TABLE 3. TRANSMITTER AND RECEIVER PARAMETER AND TIMING REGISTER BIT FORMAT

TRANSMITTER PARAMETER REG

(TPRA, TPRB)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	UNDERRUN CONTROL		IDLE	TEOM ON ZERO CNT	TX RTS CONTROL	CTS ENABLE TX	TX CHARACTER LENGTH	
COP	00 - FCS-idle 01 - reserved 10 - MARKs 11 - SYNs		0 - MARKs 1 - SYNs	OR DONE 0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes	00 - 5 bits 01 - 6 bits 10 - 7 bits 11 - 8 bits	
BOP	UNDERRUN CONTROL 00 - FCS-FLAG-idle 01 - reserved 10 - ABORT-MARKs 11 - ABORT-FLAGs		IDLE 0 - MARKs 1 - FLAGs	TEOM ON ZERO CNT OR DONE 0 - no 1 - yes				
ASYN	STOP BITS PER CHARACTER 9/16 to 1, 17/16 to 1.5, 25/16 to 2 programmable in 1/16 bit increments							

TRANSMITTER TIMING REG

(TTRA, TTRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
EXTERNAL SOURCE 0 - RTXC 1 - TRXC	TRANSMITTER CLOCK SELECT 000 - 1x external 001 - 16x external 010 - BRG 011 - DPLL 100 - own channel C/T 101 - other chan C/T			BIT RATE SELECT one of sixteen rates from BRG			

RECEIVER PARAMETER REG

(RPRA, RPRB)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYNC	not used	not used	not used	RX RTS CONTROL 0 - no 1 - yes	STRIP PARITY 0 - no 1 - yes	DCD ENABLE RX	RX CHARACTER LENGTH	
COP	SYN STRIP 0 - no 1 - yes	FCS TO FIFO 0 - no 1 - yes	AUTO HUNT & PAD CK 0 - no 1 - yes	EXT SYNC 0 - no 1 - yes	STRIP PARITY 0 - no 1 - yes	0 - no 1 - yes	00 - 5 bits 01 - 6 bits 10 - 7 bits 11 - 8 bits	
BOP	not used	FCS TO FIFO 0 - no 1 - yes	OVERRUN MODE 0 - hunt 1 - cont	EXT SYNC 0 - no 1 - yes	ALL PARTY ADDRESS 0 - no 1 - yes			

RECEIVER TIMING REG

(RTRA, RTRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
EXTERNAL SOURCE 0 - RTXC 1 - TRXC	RECEIVER CLOCK SELECT 000 - 1x external 001 - 16x external 010 - BRG 011 - C/T of channel 100 - DPLL,source=128x X/CLK 101 - DPLL,source=32x EXT 110 - DPLL,source=32x BRG 111 - DPLL,source=32x C/T			BIT RATE SELECT one of sixteen rates from BRG			

Advance Information

TABLE 3. TRANSMITTER AND RECEIVER PARAMETER AND TIMING REGISTER BIT FORMAT (Continued)

(OMRA, OMRB)

OUTPUT AND MISC REG BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
TX RESIDUAL CHARACTER LENGTH			TXRDY ACTIVATE	RXRDY ACTIVATE	OUT 2	OUT 1	RTS
000 - 1 bit							
001 - 2 bits							
010 - 3 bits			0 - FIFO not full	0 - FIFO not empty	0 - 0	0 - 0	0 - 0
011 - 4 bits							
100 - 5 bits					1 - 1	1 - 1	1 - 1
101 - 6 bits			1 - FIFO empty	1 - FIFO full			
110 - 7 bit							
111 - same as TPR[1:0]							

01 Reserved.

10 In BOP, the transmitter sends an ABORT (11111111) and then places the TXD output in a marking condition until receipt of further instructions. In COP, the transmitter places the TXD output in a marking condition until receipt of further instructions.

11 In BOP, the transmitter sends an ABORT (11111111) and then sends FLAGs until receipt of further instructions. In COP, the transmitter sends SYNs until receipt of further instructions.

[5] Idle

In BOP and COP modes, this bit selects the transmitter output during idle. Idle is defined as the state following a normal end of message until receipt of the next transmitter command.

0 Idle in marking condition.

1 Idle sending SYNs (COP) or FLAGs (BOP).

[4] Transmit EOM on Zero Count or Done

In BOP and COP modes, the assertion of this bit causes the end of message (FCS in COP, FCS-FLAG in BOP) to be transmitted upon the following events:

1. If the counter/timer is counting transmitted characters, after transmission of the character which causes the counter to reach zero count. (DONEN is also asserted as an output if the channel is in a DMA operation.)
2. If the channel is operating in DMA mode, after transmission of a character if DONEN was asserted when that character was loaded into the Tx FIFO by the DMA controller.

[7:4] Stop Bits per Character

In async mode, this field programs the length of the stop bit appended to the transmitted character as follows:

[7:4]	5 Bits/Char	6,7, or 8 Bits/Char
0000	1.063	0.563
0001	1.125	0.625
0010	1.188	0.688
0011	1.250	0.750
0100	1.313	0.813
0101	1.375	0.875
0110	1.438	0.938
0111	1.500	1.000
1000	1.563	1.563
1001	1.625	1.625
1010	1.688	1.688
1011	1.750	1.750
1100	1.813	1.813
1101	1.875	1.875
1110	1.938	1.938
1111	2.000	2.000

Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, [7] = 0 selects one stop bit and [7] = 1 selects two stop bits to be transmitted.

[3] Transmitter Request-to-Send Control

This bit controls the deactivation of the RTS_N output by the transmitter (see Detailed Operation).

0 RTS_N is not affected by status of transmitter.

1 RTS_N changes state as a function of transmitter status.

Advance Information

[2] Clear-to-Send Enable Transmitter
The state of this bit determines if the CTS_N input controls the operation of the channel's transmitter (see Detailed Operation).

0 CTS_N has no effect on the transmitter.
1 CTS_N effected by state of transmitter.

[1:0] Transmitted Bits per Character
This field selects the number of data bits per character to be transmitted. The character length does not include the start, parity, and stop bits in async or the parity bit in COP. In BOP modes the character length for the Address and Control fields is always 8 bits, and the value of this field only applies to the Information (I) field, except for the last character of the I field, whose length is specified by OMR[7:5].

Transmitter Timing Register (TTRA, TTRB)

[7] External Source
This bit selects the RTXC pin or the TRXC pin of the channel as the transmitter clock input when [6:4] specifies external. When used for input, the selected pin must be programmed as an input in the PCR [4:3] or [2:0].
0 External input from RTXC pin.
1 External input from TRXC pin.

[6:4] Transmitter Clock Select
This field selects the clock for the transmitter.

000 External clock from TRXC or RTXC at 1x the shift (baud) rate.
001 External clock from TRXC or RTXC at 16x the shift rate.
010 Internal clock from the bit rate generator at 32x the shift rate. The clock signal is divided by two before use in the transmitter which operates at 16x the baud rate. Rate selected by [3:0].
011 Internal clock from the phase locked loop at 1x the bit rate. It should be used only in half-duplex operation since the DPLL will periodically re-sync itself to the received data if in full duplex operation.
100 Internal clock from counter/timer of own channel. The C/T should be programmed to produce a clock at 32X the shift rate.
101 Internal clock from counter/timer of other channel. The C/T should be programmed to produce a clock at 32X the shift rate.
110 Reserved.
111 Reserved.

[3:0] Bit Rate Select
This field selects an output from the bit rate generator to be used by the transmitter circuits. The actual frequency output from the BRG is 32X the bit rate shown in table 4. With a crystal or external clock of 14.7456MHz the bit rates are as given in table 4 (this input is divided by two before being applied to the oscillator circuit).

TABLE 4. RECEIVER/TRANSMITTER BAUD RATES

[3:0]	Bit Rate	[3:0]	Bit Rate
0000	50	1000	1050
0001	75	1001	1200
0010	110	1010	2000
0011	134.5	1011	2400
0100	150	1100	4800
0101	200	1101	9600
0110	300	1110	19.2K
0111	600	1111	38.4K

Receiver Parameter Register (RPRA, RPRB)

[7] SYN Stripping
This bit controls the DUSCC processing in COP modes of SYN 'character patterns' that occur after the initial character synchronization. Refer to Detailed Operation for the receiver for details and definition of SYN 'patterns'.

0 Strip only SYN "patterns" required to establish character sync.
1 Strip all SYN "patterns" (including all odd DLE's in BISYNC transparent mode)

[6] Transfer Received FCS to FIFO
In BISYNC and BOP modes, the assertion of this bit causes the received FCS to be loaded into the RXFIFO. BOP mode operates correctly only if a minimum of two extra FLAGs (without shared zeros) are appended to the frame. If the FCS is specified to be transferred to the FIFO, the EOM status bit will be tagged onto the last byte of the FCS instead of to the last character of the message.
0 Do not transfer FCS to RXFIFO.
1 Transfer FCS to FIFO.

[5] Auto-Hunt and Pad Check (BISYNC)
In BISYNC mode, the assertion of this bit causes the receiver to go into hunt for character sync mode after detecting certain end-of-message (EOM) characters. These are defined in the Detailed Operations section for COP receiver operation.

Advance Information

0 Disable auto-hunt and PAD check.
1 Enable auto-hunt and PAD check.

[5] Overrun Mode (BOP)
The state of this control bit determines the operation of the receiver in the event of a data overrun, i.e., when a character is received without an empty FIFO position available.
0 The receiver terminates receiving the current frame and goes into hunt phase, looking for a FLAG to be received.
1 The receiver continues receiving the current frame. The overrunning character is lost.

[4] Receiver Request-to-Send Control
See Detailed Operation.
0 Receiver does not control RTS_N output.
1 Receiver can reset RTS_N output.

[4] External Sync
In COP and BOP modes, the assertion of this bit enables external character synchronization. Each positive signal on the DCD_N/SYNIN_ pin will cause the receiver to establish synchronization on the next rising edge of the receiver clock. Character assembly will start with the RXD input at this edge. The sync signal can then be lowered. Receipt of the external sync input causes the SYN/FLAG DETECT status bit (RSR[2]) to be set and the SYNOUT pin to be asserted for one bit time. When this mode is enabled, the internal SYN (COP, BISYNC) and FLAG (BOP) detection and special character recognition (e.g., ABORT, IDLE, EOT, ETX, etc.) circuits are disabled. In BOP mode, address comparisons are not performed and character assembly begins as if in the I-field. The receiver remains in this mode and further external syncs are neglected until the receiver is disabled.
0 External sync not enabled.
1 External sync enabled.

Note that EXT SYNC and DCD ENABLE RX cannot be asserted simultaneously since they use same pin.

[3] Strip Parity
In COP and async modes with parity enabled, this bit controls whether the received parity bit is stripped from the data placed in the receiver FIFO. It is valid only for programmed character lengths of 5, 6, and 7 bits. If the bit is stripped, the corresponding bit in the received data is set to zero.
0 Transfer parity bit as received.
1 Strip parity bit from data.

[3] All Parties Address
In BOP secondary modes, the assertion of this bit causes the receiver to 'wake up' upon receipt of the address H'FF' or H'FF, FF', for single and dual octet address modes respectively, in addition to its normal station address. This feature allows all stations to receive a message.

0 Don't recognize all parties address.
1 Recognize all parties address.

[2] DCD Enable Receiver
If this bit is asserted, the DCD_N/SYNIN_ input must be low in order for the receiver to operate. If the input is negated (goes high) while a character is being received, the receiver terminates receipt of the current message (this action in effect resets the receiver). If DCD subsequently goes high, the receiver will search for the start bit, SYN pattern, or FLAG, depending on the channel protocol. (Note that the change of input can be programmed to generate an interrupt; the duration of the DCD level change is described in the discussion of the input and counter/timer register ICTSRA/B).
0 DCD not used to enable receiver
1 DCD used to enable receiver

EXT SYNC and DCD ENABLE RX cannot be asserted simultaneously since they use same pin.

[1:0] Received Bits per Character
This field selects the number of data bits per character to be assembled by the receiver. The character length does not include the start, parity, and stop bits in async or the parity bit in COP. In BOP modes, the character length for the address and control fields is always 8 bits, and the value of this field only applies to the information field. If the number of bits assembled for the last character of the I field is less than the value programmed in this field, RCL not zero (RSR[0]) is asserted and the actual number of bits received is given in TRSR[2:0].

Receiver Timing Register (RTRA, RTRB)

[7] External Source
This bit selects the RTXC pin or the TRXC pin of the channel as the receiver or DPLL clock input, when [6:4] specifies external. When used for input, the selected pin must be programmed as an input in the PCR [4:3] or [2:0].
0 External input from RTXC pin.
1 External input from TRXC pin.

Advance Information

[6:4] Receiver Clock Select
This field selects the clock for the receiver.
000 External clock from TRXC or RTXC at 1X the shift (baud) rate.
001 External clock from TRXC or RTXC at 16X the shift rate.
010 Internal clock from the bit rate generator at 32X the shift rate. Clock is divided by two before use by the receiver logic, which operates at 16X the baud rate. Rate selected by [3:0].
011 Internal clock from counter/timer of own channel. The C/T should be programmed to produce a clock at 32X the shift rate. Clock is divided by two before use in the receiver logic.
100 Internal clock from the phase locked loop. The clock for the DPLL is a 128X clock from the crystal oscillator or system clock input. (The input to the oscillator is divided by two and the input to the DPLL from the oscillator is then divided by two.)
101 Internal clock from the phase locked loop. The clock for the DPLL is an external 32X clock from the RTXC or TRXC pin, as selected by [7].
110 Internal clock from the phase locked loop. The clock for the DPLL is a 32X clock from the BRG. The frequency is programmed by [3:0].
111 Internal clock from the phase locked loop. The clock for the DPLL is a 32X clock from the counter/timer of the channel.

[3:0] Bit Rate Select
This field selects an output from the bit rate generator to be used by the receiver circuits. The actual frequency output from the BRG is 32X the bit rate shown in table 4.

Output and Miscellaneous Register (OMRA, OMRB)

[7:5] Transmitted Residual Character Length
In BOP modes, this field determines the number of bits transmitted for the last character in the Information field. This length applies to:
- the character in the transmit FIFO accompanied by the FIFO'ed 'send EOM' command.
- the character loaded into the FIFO by the DMA controller if DONEN is simultaneously asserted and TPR[4] is asserted.
- the character loaded into the FIFO which causes the counter to reach zero count when TPR[4] is asserted.

The length of all other characters in the frame's information field is selected by TPR[1:0]. If this field is 111, the number of bits in the last character is the same as programmed in TPR[1:0].

[4] TXRDY Activate Mode
0 FIFO not full. The channel's TXRDY status bit is asserted each time a character is transferred from the transmit FIFO to the transmit shift register. If not reset by the CPU, TXRDY remains asserted until the FIFO is full, at which time it is negated.
1 FIFO empty. The channel's TXRDY status bit is asserted when a character transfer from the transmit FIFO to the transmit shift register causes the FIFO to become empty. If not reset by the CPU, TXRDY remains asserted until the FIFO is full, at which time it is negated.

[3] RXRDY Activate Mode
0 FIFO not empty. The channel's RXRDY status bit is asserted each time a character is transferred from the receive shift register to the receive FIFO. If not reset by the CPU, RXRDY remains asserted until the receive FIFO is empty, at which time it is negated.
1 FIFO full. The channel's RXRDY status bit is asserted when a character transfer from the receive shift register to the receive FIFO causes the FIFO to become full. If not reset by the CPU, RXRDY remains asserted until the FIFO is empty, at which time it is negated.

[2] General Purpose Output 2
This general purpose bit is used to control the TXDRQ_N/GPO2_/RTS_N pin, when it is used as an output. The output is high when the bit is a 0 and is low when the bit is a 1.

[1] General Purpose Output 1
This bit is used to control the RTXDRQ_N/GPO1_N output, which is a general purpose output when the channel is not in DMA mode. The output is high when the bit is a 0 and is low when the bit is a 1.

[0] Request-to-Send Output
This bit controls the TXDRQ_N/GPO2_N/RTS_N and SYNOUT_N/RTS_N pin, when either is used as a RTS output. The output is high when the bit is a 0 and is low when the bit is a 1.

COUNTER/TIMER AND VALUE REGISTERS

There are five registers in this set consisting of the following:

1. Counter/timer control register (CTCRA/B)
2. Counter/timer preset high and low registers (CTPHA/B, CTPLA/B)
3. Counter/timer (current value) high and low registers (CTHA/B, CTLA/B)

Advance Information

The format of each of the registers of this set is contained in table 5. The control register contains the operational information for the counter/timer. The preset registers contain the count which is loaded into the counter/timer circuits. The third group contains the current value of the counter/timer as it operates.

Counter/Timer Control Register (CTCRA, CTCRB)

[7] Zero Detect Interrupt
This bit determines whether the assertion of the C/T ZERO COUNT status bit (ICTSR[6]) causes an interrupt to be generated.
0 Interrupt disabled.
1 Interrupt enabled if master interrupt enable (ICR[1] or ICR[0]) is asserted.

[6] Zero Detect Control
This bit determines the action of the counter upon reaching zero count.
0 The counter/timer is preset to the value contained in the counter/timer preset registers (CTPRL, CTPRH) at the next clock edge.
1 The counter/timer continues counting without preset. The value at the next clock edge will be H'FFFF'.

[5] TRXC or RTXC Counter/Timer
This bit selects the output waveform when the counter/timer is selected to be output on TRXC or RTXC.
1 The output is a single clock positive width pulse each time the C/T reaches zero count. (The duration of this pulse is one bit time.)
0 The output toggles each time the C/T reaches zero count. The output is cleared to low by each 'start C/T' command.

[4:3] Clock Select
This field selects whether the clock selected by [2:0] is prescaled prior to being applied to the input of the C/T.
00 No prescaling.
01 Divide clock by 16.
10 Divide clock by 32.
11 Divide clock by 64.

[2:0] Clock Source
This field selects the clock source for the counter timer.

000 RTXC pin. Pin must be programmed as input.
001 TRXC pin. Pin must be programmed as input.
010 Source is the crystal oscillator or system clock input divided by four.
011 This selects a special mode of operation. In this mode the counter, after receiving the 'start C/T' command, delays the start of counting until the RXD input goes low. It continues counting until the RXD input goes high, then stops and sets the C/T zero count status bit. The CPU can use the value in the C/T to determine the bit rate of the incoming data. The clock is the crystal oscillator or system clock input divided by four.
100 Source is the 32x BRG output selected by RTR[3:0] of own channel.
101 Source is the 32x BRG output selected by TTR[3:0] of own channel.
110 Source is the internal signal which loads received characters from the receive shift register into the receiver FIFO. When operating in this mode, the FIFO'ed EOM status bit (RSR[7]) shall be set when the character which causes the count to go to zero is loaded into the receive FIFO.
111 Source is the internal signal which transfers characters from the data bus into the transmit FIFO. When operating in this mode, and if the TEOM on zero count or done control bit (TPR[4]) is asserted, the FIFO'ed Send EOM command shall be automatically asserted when the character which causes the count to go to zero is loaded into the transmit FIFO.

Counter/Timer Preset High Register (CTPRHA, CTCPRHB)

[7:0] MSB
This register contains the eight most significant bits of the value loaded into the counter/timer upon receipt of the load C/T from preset register command or when the counter/timer reaches zero count and the zero detect control bit (CTCR[6]) is negated.

Counter/Timer Preset Low Register (CTPRLA, CTPRLB)

[7:0] LSB

This register contains the eight least significant bits of the value loaded into the counter/timer upon receipt of the load C/T from preset register command or when the counter/timer reaches zero count and the zero detect control bit (CTCR[6]) is negated.

Counter/Timer High Register (CTHA, CTHB)

[7:0] MSB

A read of this 'register' provides the eight most significant bits of the current value of the counter timer. It is recommended that the C/T be stopped via a stop counter command before it is read in order to prevent errors which may occur due to the read being performed while the C/T is changing. This count is continued after the register is read.

Advance Information

TABLE 5. - COUNTER/TIMER CONTROL AND VALUE REGISTER BIT FORMATS

(CTCRA, CTCRB)

COUNTER/TIMER CONTROL REG

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ZERO DETECT INTERRUPT	ZERO DETECT CONTROL	OUTPUT CONTROL	PRESCALER		CLOCK SOURCE		
0-disable 1-enabled	0-preset 1-continu	0 -square 1 -pulse	00 - 1 01 - 16 10 - 32 11 - 64		000 - RTXC pin 001 - TRXC pin 010 - X1/CLOCK divided by 4 011 - " " gated by RXD 100 - RX BRG 101 - TX BRG 110 - RX CHARACTERS 111 - TX CHARACTERS		

(CTPRHA, CTPRHB)

COUNTER/TIMER PRESET HIGH REG

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
MOST SIGNIFICANT BITS OF COUNTER/TIMER PRESET VALUE							

(CTPRLA, CTPRLB)

COUNTER/TIMER PRESET REGISTER LOW

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LEAST SIGNIFICANT BITS OF COUNTER/TIMER PRESET VALUE							

(CTHA, CTHB)

COUNTER/TIMER HIGH

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
MOST SIGNIFICANT BITS OF COUNTER/TIMER							

(CTLA, CTLB)

COUNTER/TIMER LOW

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LEAST SIGNIFICANT BITS OF COUNTER/TIMER							

Counter/Timer Low Register (CTLA, CTLB)

[7:0] LSB
A read of this 'register' provides the eight least significant bits of the current value of the counter timer. It is recommended that the C/T be stopped via a stop counter command before it is read, in order to prevent errors which may occur due to the read being performed while the C/T is changing. This count is continued after the register is read.

INTERRUPT CONTROL AND STATUS REGISTERS

This group of registers define mechanisms for communications between the DUSCC and the processor and contain the device status information. Four registers are available for each channel and four common device registers comprise this group which consists of the following:

1. Interrupt Enable register (IERA/B)
2. Receiver status register (RSRA/B)
3. Transmitter and receiver status register (TRSRA/B)
4. Input and counter/timer status register (ICTSRA/B)
5. Interrupt vector registers (IVR)and (IVRM)
6. Interrupt control register (ICR)
7. General status register (GSR)

See table 6 for register bit formats and figure 1 for register relationships.

Advance Information

TABLE 6. INTERRUPT CONTROL AND STATUS REGISTER BIT FORMAT

RECEIVER STATUS REG

(RSRA, RSRB)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYNC	# CHAR COMPARE	RTS NEGATED	OVERRUN ERROR	not used	BRK END DETECT	BRK START DETECT	# FRAMING ERROR	# PARITY ERROR
COP	# EOM DETECT +	PAD ERROR +	OVERRUN ERROR	not used	not used	SYN DETECT	# CRC ERROR	# PARITY ERROR
BOP	# EOM DETECT	ABORT DETECT	OVERRUN ERROR	SHORT FRAME DET	IDLE DETECT	FLAG DETECT	# CRC ERROR	# RCL NOT ZERO

NOTES: '#' indicates that the status bit is FIFO'ed
'+' indicates COP BISYNC mode only

TRANSMITTER AND RECEIVER STATUS REG

(TRSRA, TRSRB)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ASYNC	TRANSMIT EMPTY	CTS UNDERRUN	not used	SEND BREAK ACK	DPLL ERROR	not used	not used	not used
COP	TRANSMIT EMPTY	CTS UNDERRUN	FRAME COMPLETE	SEND SOM ACK	DPLL ERROR	not used	RX HUNT MODE	RX XPNT MODE
BOP	TRANSMIT EMPTY	CTS UNDERRUN	FRAME COMPLETE	SEND SOM/ ABORT ACK	DPLL ERROR	RX RESIDUAL CHARACTER LENGTH 000 - 1 bit 001 - 2 bits 010 - 3 bits 011 - 4 bits		100 - 5 bits 101 - 6 bits 110 - 7 bits 111 - 0 bits

INPUT AND COUNTER/TIMER STATUS REGISTER

(ICTSRA, ICTSRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
C/T RUNNING	C/T ZERO COUNT	DELTA DCD	DELTA CTS	DCD	CTS	GPI1	GPI2

INTERRUPT ENABLE REG

(IERA, IERB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DCD/CTS	TXRDY	TRSR 73	RXRDY	RSR 76	RSR 54	RSR 32	RSR 10
0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes	0 - no 1 - yes

INTERRUPT VECTOR REG AND INTERRUPT VECTOR MODIFIED REG

(IVR), (IVRM)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
8-BIT INTERRUPT VECTOR							

Advance Information

TABLE 6. INTERRUPT CONTROL AND STATUS REGISTER BIT FORMAT (Continued)

(GSR)

GENERAL STATUS REG

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CHANNEL B				CHANNEL A			
EXTERNAL or C/T status	RX/TX status	TXRDY	RXRDY	EXTERNAL or C/T status	RX/TX status	TXRDY	RXRDY

(ICR)

INTERRUPT CONTROL REG

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
A/B INTERRUPT PRIORITY		VECTOR MODE		BITS TO MODIFY	VECTOR INCLUDES STATUS	CHANNEL A MASTR INT ENABLE	CHANNEL B MASTR INT ENABLE
		00 - vectored					
		01 - vectored					
		10 - vectored		0 - 2:0	0 - no	0 - no	0 - no
		11 - non vectored		1 - 4:2	1 - yes	1 - yes	1 - yes

Interrupt Enable Register (IERA, IERB)

This register controls whether the assertion of bits in the channel's status registers causes an interrupt to be generated. An additional condition for an interrupt to be generated is that the channel's master interrupt enable bit, ICR[0] or ICR[1], be asserted.

[7] DCD/CTS
0 Interrupt not enabled.
1 Interrupt generated if ICTSR[4] or ICTSR[5] are asserted.

[6] TXRDY
0 Interrupt not enabled.
1 Interrupt generated if TXRDY (GSR[1] or GSR[5] for channels A and B respectively) is asserted.

[5] TRSR 73
0 Interrupt not enabled.
1 Interrupt generated if bits 7, 6, 5, 4 or 3 of the TRSR are asserted.

[4] RXRDY
0 Interrupt not enabled.
1 Interrupt generated if RXRDY (GSR[0] or GSR[4] for channels A and B respectively) is asserted.

[3] RSR 76
0 Interrupt not enabled.
1 Interrupt generated if bits 7 or 6 of the RSR are asserted.

[2] RSR 54
0 Interrupt not enabled.
1 Interrupt generated if bits 5 or 4 of the RSR are asserted.

[1] RSR 32
0 Interrupt not enabled.
1 Interrupt generated if bits 3 or 2 of the RSR are asserted.

[0] RSR 10
0 Interrupt not enabled.
1 Interrupt generated if bits 1 or 0 of the RSR are asserted.

Receiver Status Register (RSRA, RSRB)

This register informs the CPU of receiver status. Bits indicated as 'not used' in a particular mode will read as zero. The logical OR of these bits is presented in GSR[2] or GSR[6] (OR'ed with the bits of TRSR) for channels A and B respectively. Unless otherwise indicated, asserted status bits are reset only by performing a write operation to the status register with the bits to be reset being ones in the accompanying data word, or when the RESETN input is asserted, or when a 'reset receiver' command is issued.

Certain status bits are specified as being FIFO'ed. This means that they occupy positions in a status FIFO that correspond to the data FIFO. As the data is brought to the top of the FIFO (the position read when the RXFIFO is read), the FIFO'ed status bits are logically OR'ed with the previous contents of the corresponding bits in the status register. This permits the user to obtain status either character by character or on a block basis. For character by character status, the SR bits should be read and then cleared before reading the character data from RXFIFO. For block status, the status register is initially cleared and then read after the message is received.

Advance Information

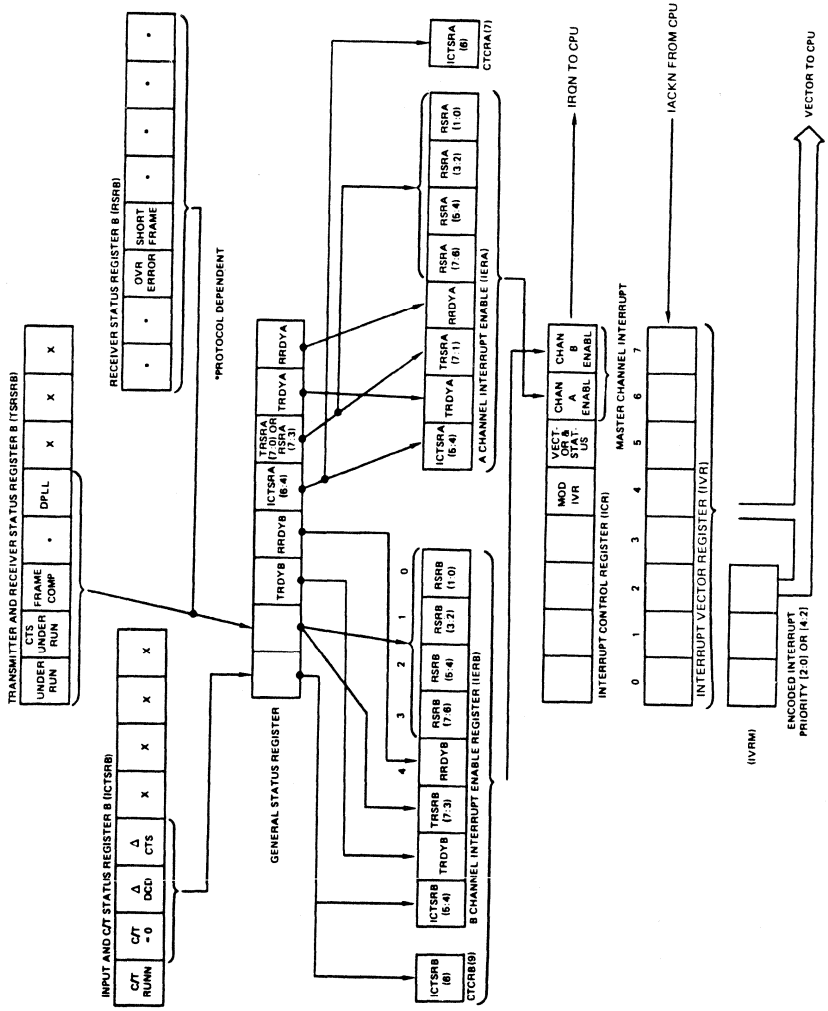


Figure 1. Interrupt Control and Status Register

Advance Information

Asserted status bits can be programmed to generate an interrupt (See Interrupt Enable Register).

[7] Character Compare (Async), EOM (BISYNC/BOP)
If the counter/timer is programmed to count received characters, this bit is asserted when the character which causes the count to go to zero is loaded into the receive FIFO. It is also asserted to indicate the following conditions:

ASYNC The character currently at the top of RXF matched the contents of SIR. A character will not compare if it is received with parity error even if the data portion matches.
BISYNC The character currently at the top of the FIFO was either a text message terminator or a control sequence received outside of a text or header field. See DETAILED OPERATION of COP Receiver. If transfer FCS to FIFO (RPR[6]) is set, the EOM will instead be tagged onto the last byte of the FCS. Note that if an overrun occurs during receipt of a message, the EOM character may be lost, but this status bit will still be asserted to indicate that an EOM was received.
BOP The character currently at the top of the FIFO was the last character of the frame.

[6] RTS Negated (Async), PAD Error (BISYNC), ABORT (BOP), ABORT/EOP
ASYNC The RTSN output was negated due to receiving the start bit of a new character while the RXFIFO was full (see RPR[4]).
BISYNC PAD error detected (see RPR[5]).
BOP An ABORT sequence consisting of a zero followed by seven ones was received after receipt of the first address octet but before receipt of the closing FLAG. The user should read RxFIFO until it is empty and determine if any valid characters from a previous frame are in the FIFO. If no character with a tagged EOM detect ([7]) is found, all characters are from the current frame and should be discarded along with any previously read by the CPU. An ABORT detect causes the receiver to automatically go into search for FLAG state.

[5] Overrun Error (All Modes)
A new character was received while the receive FIFO was full and a character was already waiting in the receive shift register to be transferred to the FIFO. The DUSCC protects the five characters previously assembled (four in RxFIFO, one in the Rx shift register) and discards the overrunning character(s).

[4] Short Frame (BOP)
ASYNC Not used
COP Not used
BOP A frame with fewer than the expected minimum number of bits was received.

[3] BREAK End Detect (Async), SYN Detect (COP), FLAG Detect (BOP)
ASYNC The RXD input has returned to the marking (high) state for at least one-half bit time after detecting a break. For 1X clock mode, a half bit time is defined as the time between any two edges of the input clock. For 16X mode, a half bit time is defined as eight clock cycles.
COP A SYN pattern was received. Refer to Detailed Operation for definition of SYN patterns.
BOP A FLAG sequence (01111110) was received.

[2] BREAK Start Detect (Async), IDLE (BOP)
ASYNC An all zero character, including parity (if specified) and first stop bit, was received. The receiver shall be capable of detecting breaks which begin in the middle of a previous character. Only a single all-zero character with appended framing error status bit shall be put into the FIFO when a break is detected. Additional entries to the FIFO are inhibited until the end of break has been detected (see below) and a new character is received.
COP Not used
BOP An IDLE sequence consisting of a zero followed by fifteen ones was received.

[1] Framing Error (Async), CRC Error (COP/BOP)
ASYNC At the first stop bit position the RXD input was in the low (space) state. The receiver only checks for framing error at the nominal center of the first stop bit regardless of the number of stop bits programmed in TPR[7:4].
COP In BISYNC COP mode, this bit is set upon receipt of the BCC byte(s), if any, to indicate that the received BCC was in error. The bit is normally FIFO'ed with the last byte of the frame (the character preceding the first BCC byte). However, if transfer FCS to FIFO (RPR[7]) is asserted, this bit is FIFO'ed with the last BCC byte. The value of this bit should be ignored for non-text messages or if the received frame was aborted via an ENQ. In non-BISYNC COP modes, the bit is set with

Advance Information

each received character if the current value of the CRC checker is not equal to the non-error value (see CMR2[2:0]).

BOP This bit is set upon receipt of the FCS byte(s), if any, to indicate that the received FCS was in error. The bit is normally FIFO'ed with the last byte of the I field (the character preceding the first FCS byte). However, if transfer FCS to FIFO (RPR[7]) is asserted, this bit is FIFO'ed with the last FCS byte.

[0] Parity Error (Async/COP),RCL not Zero (BOP)

ASYNC The parity bit of the received character was not as expected. A parity error does not affect the parity bit put into the FIFO as part of the character when strip parity (RPR[3]) is negated.

COP The parity bit of the received character was not as expected. A parity error does not affect the parity bit put into the FIFO as part of the character when strip parity (RPR[3]) is negated. A SYN or other character received with parity error is treated as a data character. Thus, a SYN with parity error received while in SYN search state will not establish character sync. Characters received with parity error while in the SYN search state will not set the error bit.

BOP The last character of the I field did not have the character length specified in RPR[1:0]. The actual received character length of this byte can be read in TRSR[2:0]. This bit is FIFO'ed with the EOM character but TRSR[2:0] is not. An exception occurs if the command to transfer the FCS to the FIFO is active, the bit will be FIFO'ed with the last byte of the FCS, i.e., with REOM. In the event that residual characters from two consecutive frames are received and are both in the FIFO, the length in TRSR[2:0] applies to the last received residual character.

Transmitter/Receiver Status Register (TRSRA, TRSRB)

This register informs the CPU of transmitter and receiver status. Bits indicated as not used in a particular mode will read as zero. The logical OR of bits [7:3] is presented in GSR[2] or GSR[6] (OR'ed with the bits of RSR) for channels A and B respectively. Unless otherwise indicated, asserted status bits are reset only:

1. by performing a write operation to the status register with the bits to be reset being ones in the accompanying data word.
2. when the RESETN input is asserted.
3. for [7:4], when a 'reset transmitter' command is issued.
4. for [3:0], when a 'reset receiver' command is issued.

Asserted status bits in [7:3] can be programmed to generate an interrupt. See IER.

[7] Transmitter Empty

Indicates that the transmit shift register has completed serializing a character and found no other character to serialize in the Tx FIFO. The bit is not set until at least one character from the transmit FIFO (not including PAD characters in synchronous modes) has been serialized. The transmitter action after transmitter empty depends on operating mode:

ASYNC The TXD output is held in the MARK state until another character is loaded into the TXFIFO. Normal operation then continues.

COP Action is specified by TPR[7:6].

BOP Action is specified by TPR[7:6].

[6] CTS Underrun (Async/COP/BOP)

This bit may be set only if CTS enable TX (TPR[2]) is asserted. It indicates that the transmit shift register was ready to begin serializing a character and found the $\overline{CTS}$ input negated (see Detailed Operation of Transmitter).

[5] Frame Complete (COP/BOP)

ASYNC Not used.

COP Asserted at the beginning of transmission of the end of message sequence, which normally consists of the CRC/LRC followed by SYN (COP) or FLAG (BOP).

BOP Transmission of the EOM is initiated by command or automatically under certain conditions. The CPU can invoke the send SOM command after this bit is set to control the number of SYNs/FLAGs between transmitted frames.

[4] Send Break Ack (ASYNC)/Send SOM ACK (COP)/Send SOM-Abort Ack (BOP)

ASYNC Set when the transmitter begins transmission of a break in response to the send break command. If the command is reinvoked, the bit will be set again at the beginning of the next character time. The user can control the length of the break by counting character times through this mechanism.

COP Set when the transmitter begins transmission of a SYN in response to the send SOM or send SOM with PAD command. If the command is reinvoked, the bit will be set again at the beginning of the next transmitted SYN. The user can

Advance Information

control the number of SYNs which are sent through this mechanism.

BOP Set when the transmitter begins transmission of a FLAG/ABORT in response to the send SOM or send SOM with PAD or send ABORT command. If the command is re-invoked, the bit will be set again at the beginning of the next transmitted FLAG/ABORT. The user can control the number of FLAGs/ABORTs which are sent through this mechanism.

[3] DPLL Error
Set while the DPLL is operating in FM mode to indicate that a data transition was not detected within the detection window for two consecutive bits and that the DPLL was forced into search mode. This feature is disabled when the DPLL is specified as the clock source for the transmitter via TTR[6:4].

[2:0] Received Residual Character Length (BOP)
BOP This field should be examined to determine the length of the last character of the I field (character tagged with REOM status bit) if RSR[0] is set to indicate that the length was not equal to the character length specified in RPR[1:0].

[1] Receiver in Hunt Mode (COP)
COP Indicates that the receiver is in the hunt mode, searching the data stream for a SYN sequence to establish character synchronization. The bit is negated automatically when character sync is achieved.

[0] Receiver in Transparent Mode (BISYNC)
COP Indicates that a DLE-STX sequence was received and the receiver is operating in BISYNC transparent mode. Transparent mode operation is terminated and the bit is negated automatically when one of the terminators for transparent text mode is received (DLE-ETX/ETB/ITB/ENQ).

Input and Counter/Timer Status Register (ICTSRA, ICTSRB)

This register informs the CPU of status of the counter/timer and inputs. The logical OR of bits [6:4] is presented in GSR[3] or GSR[7] for channels A and B respectively. Unless otherwise specified, bits of this register are reset only:

1. By performing a write operation to the status register with the bits to be reset (ones in the accompanying data word for bits [6:4] only).
2. When the RESETN input is asserted (bits [7:4]) only.

[7] Counter/Timing Running
Set when the C/T is started by start C/T command and reset when it is stopped by a stop C/T command.

[6] Counter/Timer Zero Detect
Set when the counter/timer reaches zero count. The assertion of this bit causes an interrupt to be generated if ICTCR[7] and the channel's master interrupt enable (ICR[1] or ICR[0]) are asserted.

[5] Delta DCD
The DCD input is sampled approximately every 6.8usec using the 32X, 4800 baud output from the BRG. This bit is set if the input changes state between any two successive samples. The reset circuitry should initialize the sampling circuits so that a change is not falsely indicated at power on time. The assertion of this bit causes an interrupt to be generated if IER[7] and the channel's master interrupt enable (ICR[1] or ICR[0]) are asserted.

[4] Delta CTS
The CTS input is sampled approximately every 6.8usec using the 32X, 4800 baud output from the BRG. This bit is set if the input changes state between any two successive samples. The reset circuitry should initialize the sampling circuits so that a change is not falsely indicated at power on time. The assertion of this bit causes an interrupt to be generated if IER[7] and the channel's master interrupt enable (ICR[1] or ICR[0]) are asserted.

[3:0] Current State of DCD, CTS, GPI1, and GPI2 Inputs
This field provides the current state of the channel's input pins. The bit's value is latched at the beginning of the read cycle.

Interrupt Vector Register (IVR) and Modified Vector Register (IVRM)

[7:0] Register Content
If ICR[2] = 0, the content of IVR register is output on the data bus when the DUSCC has issued an interrupt request and the responding interrupt acknowledge (IACKN) is received. The value in the IVR is initialized to H'OF' on power on reset. If 'vector includes' status is specified by ICR[2] = 1, bits [2:0] or [4:2] (depending on ICR[3]), the vector is modified as shown below to indicate the highest priority interrupt currently active. The priority is programmable through the ICR. This modified vector is stored in the IVRM. When ICR[2] = 1, the content of the IVRM is output on to the data bus on the interrupt acknowledge. The vector is

Advance Information

not modified, regardless of the value of ICR[2], if the CPU has not written an initial vector into this register.

[2:0]/[4:2]	Highest Priority Interrupt Condition
000	Channel A receiver ready
001	Channel A transmitter ready
010	Channel A Rx/Tx status
011	Channel A external or C/T status
100	Channel B receiver ready
101	Channel B transmitter ready
110	Channel B Rx/Tx status
111	Channel B external or C/T status

Either the modified or unmodified vector can also be read by the CPU via a normal bus read cycle (see table 1). The vector value is locked at the beginning of the IACK or read cycle until the cycle is completed.

Interrupt Control Register (ICR)

[7:6] A/B Interrupt Priority
Selects the relative priority between channels A and B. The state of this bit determines the value of the interrupt vector (see Interrupt Vector Register). The priority within each channel, from highest to lowest, is as follows:

0 Receiver ready
1 Transmitter ready
2 Rx/Tx status
3 External or C/T status

00 Channel A has the highest priority. The DUSCC interrupt priorities from highest to lowest are as follows: (0), A(1), A(2), A(3), B(0), B(1), B(2), B(3)
10 Priorities are interleaved between channels, but channel A has the highest priority between events of equal channel priority. The DUSCC interrupt priorities from highest to lowest are as follows: A(0), B(0), A(1), B(1), A(2), B(2), A(3), B(3)
01 Channel B has the highest priority. The DUSCC interrupt priorities from highest to lowest are as follows: B(0), B(1), B(2), B(3), A(0), A(1), A(2), A(3)
11 Priorities are interleaved between channels, but channel B has the highest priority between events of equal channel priority. The DUSCC interrupt priorities from highest to lowest are as follows: B(0), A(0), B(1), A(1), B(2), A(2), B(3), A(3)

[5:4] Vector Mode
The value of this field determines the response of the DUSCC when the interrupt acknowledge (IACKN) is received from the CPU.

00 or 01 or 10 Vectored mode. Upon interrupt acknowledge, the DUSCC locks its current interrupt status until the end of the acknowledge cycle. If it has an active interrupt pending, it responds with the appropriate vector and then asserts DTACKN. If it does not have an interrupt, it propagates the acknowledge through its X2/IDCN output if this function is programmed in PCRA[7]. Otherwise, the IACKN is ignored. Locking the interrupt status at the leading edge of IACKN prevents a device at a high position in the interrupt daisy chain from responding to an IACK issued for a lower priority device while the acknowledge is being propagated to that device.
11 Nonvectored mode. The DUSCC ignores an IACK if one is received; the interrupt vector is not placed on the data bus. The internal interrupt status is locked when a read of the IVR is performed. Except for the absence of the vector on the bus, the DUSCC performs as it does in vectored mode - the vector is prioritized and modified if programmed.

[3] Vector Bits to Modify
Selects which bits of the vector stored in the IVR are to be modified to indicate the highest priority interrupt pending in the DUSCC. See Interrupt Vector Register.
0 Modify bits 2:0 of the vector.
1 Modify bits 4:2 of the vector.

[2] Vector Includes Status
Selects whether the modified (includes status) (IVRM) or unmodified vector (IVR) is output in response to an interrupt acknowledge(see Interrupt Vector Register).

0 Unmodified vector.
1 Modified vector.

[1] Channel A Master Interrupt Enable
0 Channel A interrupts are disabled.
1 Channel A interrupts are enabled.

[0] Channel B Master Interrupt Enable
0 Channel B interrupts are disabled.
1 Channel B interrupts are enabled.

General Status Register (GSR)

This register provides a 'quick look' at the overall status of both channels of the DUSCC. A write to this register with 1s at the corresponding bit positions causes TXRDY (bits 5 and 1) and/or RXRDY (bits 4 and 0) to be reset. The other status bits can be reset only by resetting the individual status bits that they point to.

Advance Information

[7] Channel B External or Counter/Timer Status
This bit indicates that one of the following status bits is asserted:
ICTSRB[6:4]

[6] Channel B Receiver or Transmitter Status
This bit indicates that one of the following status bits is asserted:
RSRB[7:0], TRSRB[7:3]

[5] Channel B Transmitter Ready
The assertion of this bit indicates that one or more characters may be loaded into the channel B transmitter FIFO to be serialized by the transmit shift register.

[4] Channel B Receiver Ready
The assertion of this bit indicates that one or more characters are available in the channel B receiver FIFO to be read by the CPU. RXRDY is initially reset (negated) by a chip reset or when a 'reset channel B receiver' command is invoked.

[3] Channel A External or Counter/Timer Status
This bit indicates that one of the following status bits is asserted:
ICTSRA[6:4]

[2] Channel A Receiver or Transmitter Status
This bit indicates that one of the following status bits is asserted:
RSRA[7:0], TRSRA[7:3]

[1] Channel A Transmitter Ready
The assertion of this bit indicates that one or more characters may be loaded into the channel A transmitter FIFO to be serialized by the transmit shift register.

[0] Channel A Receiver Ready
The assertion of this bit indicates that one or more characters are available in the channel A receiver FIFO to be read by the CPU.

Channel Command Register (CCRA, CCRB)

Commands to the DUSCC are entered through the channel command register, the format of that register is shown in table 7. A read of this register returns the last invoked command (with bits 4 and 5 set to 1).

Transmitter Commands

0000 Reset transmitter. Causes the transmitter to cease operation immediately. The transmit FIFO is cleared and the TXD output goes into the marking state. Also clears the transmitter status bits (TRSR[7:4]) and sets the TXRDY status bit (GSR[1] or GSR[5] for channels A and B respectively). Other registers are not affected.

0001 Reset transmit CRC. This command is appended to and FIFO'ed along with the next character loaded into the transmit FIFO. It causes the transmitter CRC generator to be reset to its initial state prior to beginning transmission of the appended character.

0010 Enable transmitter. Enables transmitter operation, conditioned by the state of the CTS ENABLE TX bit, TPR[2]. Has no effect if invoked when the transmitter has previously been enabled.

0011 Disable transmitter. Terminates transmitter operation and places the TXD output in the marking state at the next occurrence of a transmit FIFO empty condition. All characters currently in the FIFO, or any loaded subsequently prior to attaining an empty condition, will be transmitted.

0100 Send start of message. Used in COP and BOP modes to initiate transmission of a frame after the transmitter is first enabled, prior to sending the contents of the FIFO. Can also be used to precisely control the number of SYN/FLAGs at the beginning of transmission or in between frames.

When the transmitter is first enabled, transmission will not begin until this command (or the send SOM with PAD command, see below) is issued. The command causes the SYN (COP) or FLAG (BOP) pattern to be transmitted. SEND SOM ACK (TRSR[4]) is set when transmission of the SYN/FLAG begins. The CPU may then reinvoke the command if multiple SYN/FLAGs are to be transmitted. Transmission of the FIFO characters begins when the command is no longer reinvoked. If the FIFO is empty, SYN/FLAGs continue to be transmitted until a character is loaded into the FIFO, but the status bit is not set.

Insertion of SYN/FLAGs between frames can be accomplished by invoking this command after the frame complete status bit (TRSR[5]) has been asserted in response to transmission of the end-of-message sequence.

0101 Send start of message with opening PAD. Used in COP and BOP modes after the transmitter is first enabled to send a bit pattern for PLL synchronization prior to transmitting the opening SYN (COP) or FLAG (BOP). The SYN/FLAG is sent at the next occurrence of a transmit FIFO empty condition. All characters currently in the FIFO, or any loaded subsequently prior to attaining an empty condition, will be transmitted. While the PAD characters are

Advance Information

TABLE 7. COMMAND REGISTER BIT FORMAT

CHANNEL COMMAND REG

(CCRA, CCRB)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
00 = TRANSMITTER CMD		don't care	don't care	TRANSMITTER COMMAND 0000 - reset TX 0001 - reset TX CRC 0010 - enable TX 0011 - disable TX 0100 - send SOM 0101 - send SOM with PAD 0110 - send EOM 0111 - send BREAK/ABORT 1000 - send DLE 1001 - reserved 1010 - reserved 1011 - reserved 1100 - reserved 1101 - exclude from CRC			
01 = RECEIVER CMD		don't care	don't care	RECEIVER COMMAND 0000 - reset RX 0001 - reserved 0010 - enable RX 0011 - disable RX			
10 = C/T CMD		don't care	don't care	COUNTER/TIMER COMMAND 0000 - start 0001 - stop 0010 - preset to FFFF 0011 - preset from CTPRH/CTPRL			
11 = DPLL CMD		don't care	don't care	DPLL COMMAND 0000 - enter search mode 0001 - disable PLL 0010 - set FM mode 0011 - set NRZI mode 0100 - reserved for test 0101 - reserved for test			

transmitted, the character length is set to 8 bits, parity generation (COP) and zero insertion (BOP) are disabled . SEND SOM ACK (TRSR[4]) is set when transmission of the SYN/FLAG begins. The CPU may then invoke the SEND SOM command if multiple SYN/FLAGs are to be transmitted.

0110 Send end of message. This command is appended to the next character loaded into the transmit FIFO. It causes the transmitter to send the end-of-message sequence (selected FCS in COP modes, FCS-FLAG in BOP modes) after the appended character is transmitted. Frame complete (TRSR[5]) is set when transmission of the FCS begins. This command is also asserted automatically if the TEOM on zero count or done control bit (TPR[4]) is asserted, and the counter/timer is programmed to count transmitted characters when the character which causes the count to go to zero is loaded into the transmit FIFO.

0111 Send Break (Async)/ Send Abort (BOP). In async mode, causes a break (space) to be transmitted after transmission of the character currently in the shift register is completed. Send break ack (TRSR[4]) is set when the transmission of the break begins. The transmitter keeps track of character times. If the command is reasserted, send break ack will be set again at the beginning of the next character time. The user can use this mechanism to control the length of the break in character time multiples. Transmission of the break is terminated by issuing a 'reset Tx' or 'disable Tx' command.

Advance Information

In BOP modes, causes an abort (eight ones) to be transmitted after transmission of the character currently in the shift register is completed. The transmitter then sends MARKs or FLAGs depending on the state of underrun control (TPR[7:6]). Send SOM/abort ack (TRSR[4]) is set when the transmission of the abort begins. If the command is reasserted before transmission of the previous ABORT is completed, the process will be repeated. This can be used to send the idle sequence. The 'send SOM' command must be used to initiate transmission of a new message. In either mode, invoking this command causes the transmit FIFO to be flushed (characters are not transmitted).

1000 Send DLE. Used in COP modes only. This command is appended to and FIFO'ed with the next character loaded into the transmitter FIFO. It causes the transmitter to send a DLE, (EBCDIC H'10', ASCII H'90') prior to transmitting the appended character. If the transmitter is operating in BISYNC transparent mode, the command automatically causes a second DLE to be loaded whenever a DLE is loaded into the FIFO. An extra (third) DLE, however, will not be sent if the send DLE command is invoked.

1001 Reserved
1010 Reserved
1011 Reserved
1100 Reserved
1101 Exclude from CRC. This command is appended to and FIFO'ed along with the next character loaded into the transmit FIFO. It causes the transmitter CRC generator to be disabled while the appended character is being transmitted. Thus, that character is not included in the CRC accumulation.

Receiver Commands

0000 Reset Receiver. Causes the receiver to cease operation, clears the receiver FIFO, and clears the receiver status (RSR[7:0], TRSR[3:0], and either GSR[0] or GSR[4] for channels A and B respectively). Other registers are not affected.
0001 Reserved.
0010 Enable receiver. Causes receiver operation to begin, conditioned by the state of the DCD ENABLE RX bit, RPR[2]. Receiver goes into START, SYN, or FLAG search mode depending on mode. Has no effect if invoked when the receiver has previously been enabled.
0011 Disable receiver. Terminates operation of the receiver. Any character currently being assembled will be lost. Does not affect FIFO or any status.

Counter/Timer Commands

0000 Start. Starts the counter/timer and prescaler.
0001 Stop. Stops the counter timer and prescaler. Since the command may be asynchronous with the selected clock source, the counter/timer and/or prescaler may count one or more additional cycles before stopping.
0010 Preset to FFFF. Presets the counter timer to H'FFFF' and the prescaler to its initial value.
0011 Preset from CTPRH/CTPRL. Transfers the current value in the counter/timer presets registers to the counter/timer and presets the prescaler to its initial value.

Digital Phase Locked Loop Commands

0000 Enter Search Mode. In NRZI mode, this command causes the DPLL counter to be set to the value 15 and the clock output will be forced high. The counter will be disabled until a transition on the data line is detected, at which point it will start incrementing and the clock output will go from high to low. After the counter reaches a count of 31, it will reset to zero and cause the clock output to go from low to high. The DPLL will then continue normal operation. This allows the DPLL to be locked onto the data without pre-frame transitions. In FM mode, this command causes the DPLL counter to be set to zero and the DPLL output clock to go high. The counter will be disabled until a transition on the data line is detected, at which point the DPLL will start normal operation.

This command should not be used if the DPLL is programmed to supply the clock for the transmitter and the transmitter is active.

0001 Disable PLL. Disables operation of the DPLL.
0010 Set FM Mode. Sets the DPLL to the FM mode of operation, used when FM0, FM1, or Manchester (NRZ) is selected by CMR1[7:6] and starts operation of the DPLL.
0011 Set NRZI Mode. Sets the DPLL to the NRZI mode of operation, used when NRZ or NRZI is selected by CMR1[7:6], and starts operation of the DPLL.
0100 Reserved for test
0101 Reserved for test

DETAILED OPERATION

INTERRUPT CONTROL

A single interrupt output (IRQN) is provided which is activated upon the occurrence of any of the following conditions:

Advance Information

Channel A external or C/T special condition
Channel B external or C/T special condition
Channel A Rx/Tx error or special condition
Channel B Rx/Tx error or special condition
Channel A TxRDY
Channel B TxRDY
Channel A RxRDY
Channel B RxRDY

Each of the above condition occupies a bit in the general status register (GSR). If ICR[2] is set, the eight conditions are encoded into three bits which are inserted with bits [2:0] or [4:2] of the interrupt vector register. This forms the content of the IVRM during an interrupt acknowledge cycle. Unmodified and modified vectors can be read directly through specified registers. Two of the conditions are the inclusive OR of several other maskable conditions:

- Ext or C/T special condition: DELTA DCD, DELTA CTS or C/T ZERO COUNT (ICTSR[6:4]).
- Rx/Tx error or special condition: any condition in the receiver status register (RSR[7:0]) or a transmitter or DPLL condition in the transmitter and receiver status register (TRSR[7:3]).

The TxRDY and RxRDY conditions are defined by OMR[4] and OMR[3], respectively. Also associated with the interrupt system are the interrupt enable register (IER), one bit in the counter/timer control register (CTCR), and the interrupt control register (ICR).

The IER is programmed to enable specified conditions or groups of conditions to cause an interrupt by asserting the corresponding bit. A negated bit prevents an interrupt from occurring when the condition is active and hence masks the interrupt. In addition to the IER, CTCR[7] can be programmed to enable or disable an interrupt upon the C/T zero count condition.

The interrupt priorities within a channel are fixed. Priority between channels is controlled by ICR[7:6]. Refer to the ICR [7:6] description.

The ICR contains the master interrupt enables for each channel (ICR[1] and ICR[0]) which must be set if the corresponding channel is to cause an interrupt. The CPU vector mode is specified by ICR[5:4] which selects either vectored or non-vectored operation. If vectored mode is selected,, the content of the IVR or IVRM is placed on the data bus when IACK is activated. If ICR[2] is set, the content of IVRM is output which contains the content of the IVR and the encoded status of the interrupting condition.

Upon receiving an interrupt acknowledge, the DUSCC locks its current interrupt status until the end of the acknowledge cycle. If it has anactive interrupt pending, it responds with the appropriate vector and then asserts DTACKN. If it does not have an interrupt, it propagates the acknowledge through its X2/IDCN output if this function is programmed in PCRA[7]; otherwise, the IACKN is ignored. Locking the interrupt status at the leading edge of IACKN prevents a device at a high position in the interrupt daisy chain from responding to an IACK issued for a lower priority device while the acknowledge is being propagated to that device.

DMA CONTROL

The DMA control section provides the interface to allow the DUSCC to operate with an external DMA controller. One of four modes of DMA can be programmed for each channel independently via CMR2[5:3]:

- Half duplex single address. In this mode, a single pin provides both DMA read and write requests. Acknowledgement of the requests is via a single DMA acknowledge pin. The data transfer is accomplished in a single bus cycle - the DMA controller places the memory address of the source or destination of the data on the address bus and then issues the acknowledge signal, which causes the DUSCC to either write the data into its transmit FIFO (write request) or to output the contents of the top of the receive FIFO (read request). The cycle is completed when the DTCN input is asserted by the DMA controller. This mode is available in the 48-pin version only and can be used when channel operation is half duplex (e.g., BISYNC). It allows a single DMA channel to service the receiver and transmitter.
- Half duplex dual address. In this mode, a single pin provides both DMA read and write requests. Acknowledgement of the requests is via normal bus read and write cycles. The data transfer requires two bus cycles - the DMA controller acquires the data from the source (memory for a Tx DMA or DUSCC for a Rx DMA) on the first cycle and deposits it at the destination (DUSCC for a Tx DMA or memory for a Rx DMA) on the second bus cycle. This mode is used when channel operation is half duplex (e.g., BISYNC) and allows a single DMA channel to service the receiver and transmitter.
- Full duplex single address. This mode is similar to half duplex single address mode but provides separate request and acknowledge pins for the receiver and transmitter. It is available only in the 48-pin version.
- Full duplex dual address. This mode is similar to half duplex single address mode but provides separate request pins for the receiver and transmitter.

Advance Information

Figures 2 through 5 describe operation of the DUSCC in the various DMA environments.

Table 8 summarizes pins used for the DMA request and acknowledge function for the transmitter and receiver for the different DMA modes.

The DMA request signals are functionally identical to the TXRDY and RXRDY status signals for each serial channel except that in DMA the signals are negated on the leading edge of the acknowledge signal when the subsequent transfer causes the FIFO to become full (transmitter request) or empty (receiver request). In non DMA operation TXRDY and RXRDY signals are negated only after the transfer is completed. The DMA read request can be programmed to be asserted either when any character is in the receive FIFO or only when the receive FIFO is full. Likewise, the DMA write request can be programmed to be asserted either when the transmit FIFO is not full or only when the transmit FIFO is empty (the transmitter must be enabled for a DMA request to be asserted). The request signals are negated when the respective data transfer cycle is completed. When the serial channel is not operating in DMA mode, the request and acknowledge pins for the channel can be programmed for other functions (see pin descriptions).

DMA DONEN OPERATION

As an input, DONEN is asserted by the DMA controller concurrent with the corresponding DMA acknowledge to indicate to the DUSCC that the character being transferred into the Tx FIFO is the last character of the transmission frame. In synchronous modes, the DUSCC can be programmed to automatically transmit the frame termination sequence (e.g., FCS-FLAG in BOP mode) upon receipt of this signal.

As an output, DONEN is asserted by the DUSCC under the following conditions:

a. In response to the DMA acknowledge for a receiver DMA request if the FIFO'ed RECEIVED EOM status bit (RSR[7]) is set for the character being transferred.
b. In response to the DMA acknowledge for a transmitter DMA request if the counter/timer has been programmed to count transmitted characters and the terminal count has occurred.

BLOCK TRANSFERS USING DTACK

The DTACKN line may be used to synchronize data transfers to and from the DUSCC utilizing a 'wait' state. Either the receiver or the transmitter or both may be programmed for this mode of operation, independently for each channel, via CMR2[5:3].

In this mode, if the CPU attempts a write to the transmit FIFO and an empty FIFO position is not available, the DTACK line will remain negated until a position empties. The data will then be written into the FIFO and DTACKN will be asserted to signify that the transfer is complete.

Similarly, a read of an empty receive FIFO will be held off until data is available to be transferred. Potentially, this mode can cause the microcomputer system to hang up if, for example, a read request was made and no further data was available.

TIMING CIRCUITS

The timing block for each channel consists of a crystal oscillator, a bit rate generator (BRG), a digital phase locked loop (DPLL) and a 16-bit counter/timer (C/T) (see figure 6).

Crystal Oscillator

The crystal oscillator operates directly from a crystal (normally 14.7456MHz if the internal BRG is to be used) connected across the X1/CLK and X2/IDCN pins with a minimum of external components. If an external clock of the appropriate frequency is available, it may be connected to the X1/CLK pin. This signal is divided by two to provide the internal system clock (a maximum of 16MHz input is allowed).

Bit Rate Generator

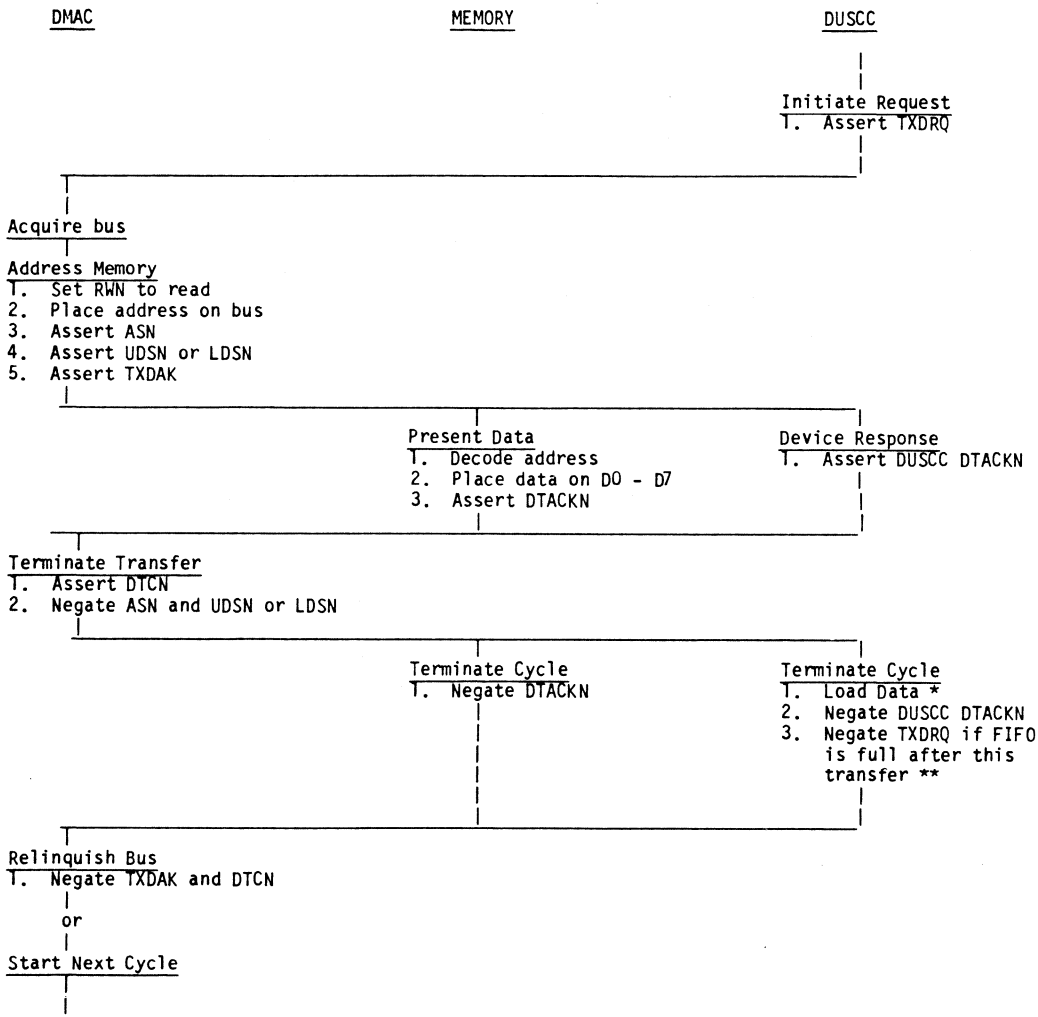
The BRG operates from the oscillator or external clock and is capable of generating 16 bit rates. These are available to the receiver, transmitter, DPLL, and C/T. The BRG output is at 32X the base bit rate. Since all sixteen rates are generated simultaneously, each receiver and transmitter may select its bit rate independently. The transmitter and receiver timing registers include a 4-bit field for this purpose (TTR[3:0], RTR[3:0]).

Digital Phase Locked Loop

Each channel of the DUSCC includes a DPLL used in synchronous modes to recover clock information from a received data stream. The DPLL is driven by a clock at nominally 32 times the data rate. This clock can be programmed, via RTR (7:4), to be supplied from an external input, from the receiver BRG, from the C/T, or directly from the crystal oscillator.

The DPLL uses this clock, along with the data stream to construct a data clock which may then be used as the DUSCC receive clock, transmit clock, or both. The output of the DPLL is a square wave at 1X the data rate. The derived clock can also be programmed to be output on a DUSCC pin; only the DPLL receiver output clock is available at the TRXC pin.

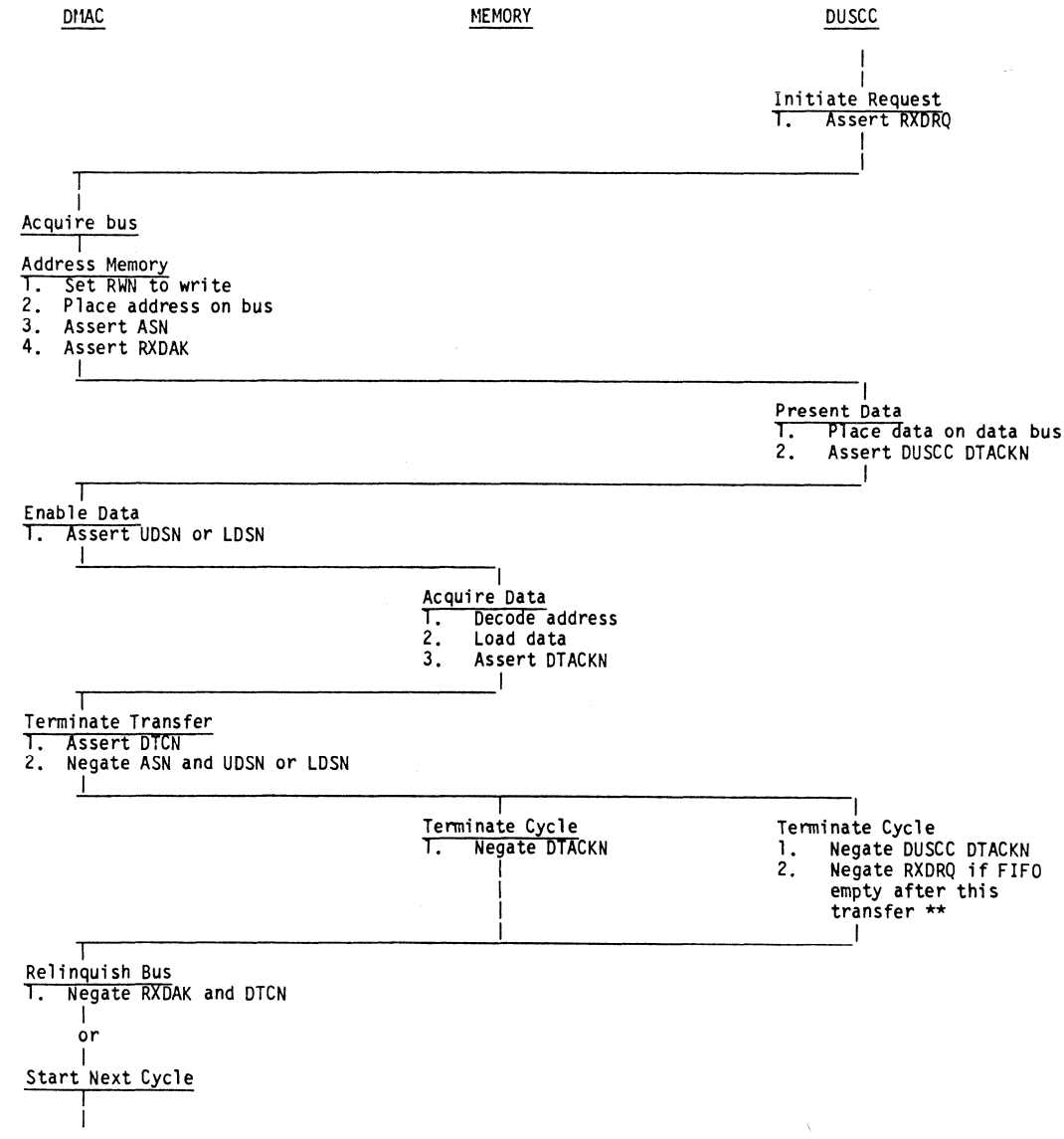
Advance Information



* On falling edge of DTCN
** On falling edge of TXDAK

Figure 2. Transmitter DMA Request Operation - Single Address Mode

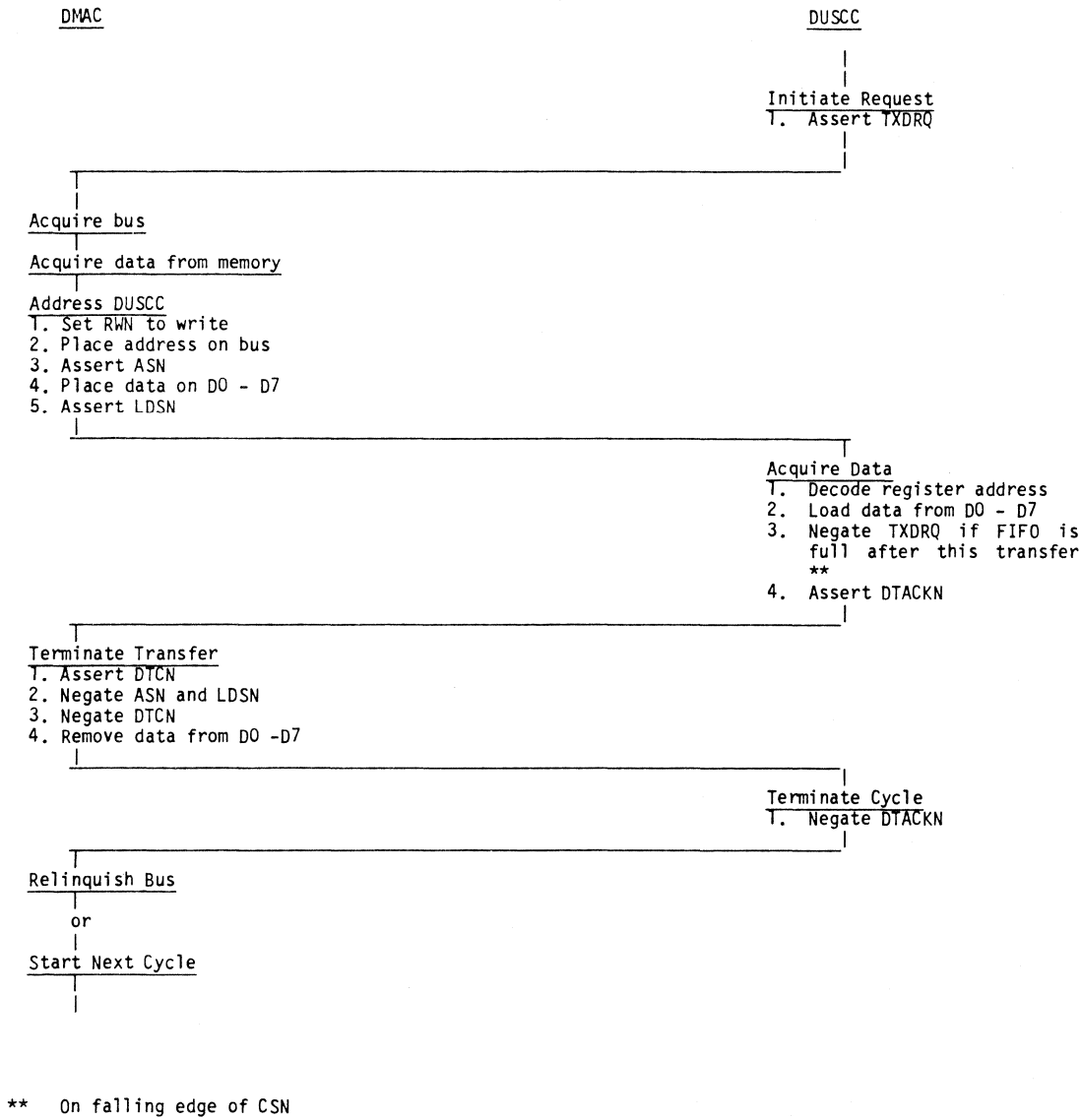
Advance Information



** On falling edge of RXDAK

Figure 3. Receiver DMA Request Operation - Single Address Mode

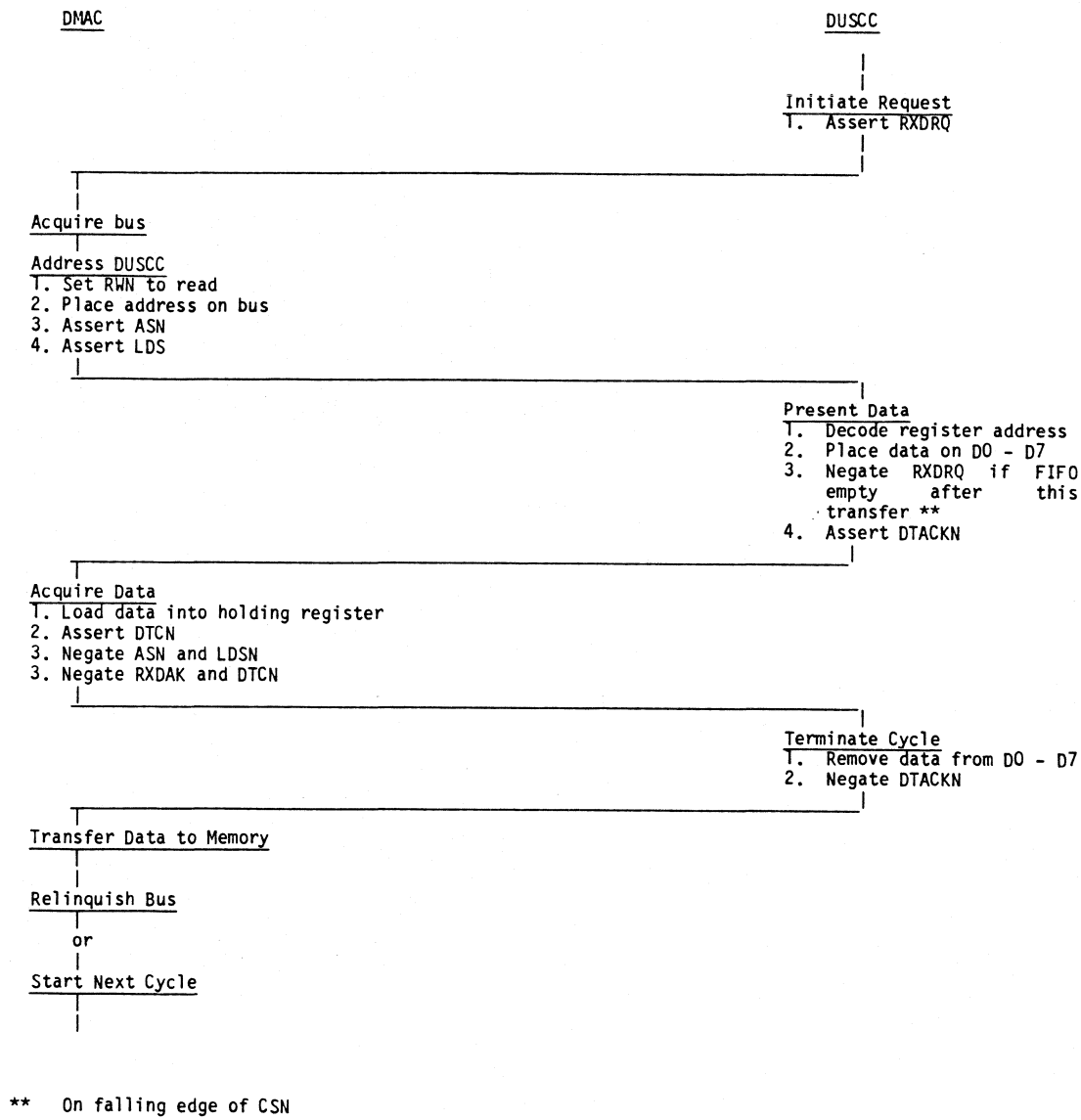
Advance Information



** On falling edge of CSN

Figure 4. Transmitter DMA Request Operation - Dual Address Mode

Advance Information



** On falling edge of CSN

Figure 5. Receiver DMA Request Operation - Dual Address Mode

Advance Information

TABLE 8. DMA REQ AND ACK FOR OPERATIONAL MODES

Function	Half Duplex Single Addr DMA	Half Duplex Dual Addr DMA	Full Duplex Single Addr DMA	Full Duplex Dual Addr DMA
RCVR REQ	RTXDRQ_N	RTXDRQ_N	RTXDRQ_N	RTXDRQ_N
TRAN REQ	same as RCVR REQ	same as RCVR REQ	TXDRQ_N	TXDRQ_N
RCVR ACK	RTXDAK_N	normal read RCVR FIFO	RTXDAK_N	normal read RCVR FIFO
TRAN ACK	same as RCVR ACK	normal write TRAN FIFO	TXDAK_N	normal write TRAN FIFO

Four commands are associated with DPLL operation: Enter search mode, set FM mode, set NRZI mode, and disable DPLL. The commands are described in the command register description. Waveforms associated with the DPLL are illustrated in figure 7.

NRZI MODE OPERATION

This mode is used with NRZ and NRZI data encoding. With this type of encoding, the transitions of the data stream occur at the beginning of the bit cell. The DPLL has a six bit counter which is incremented by a 32X clock. The first edge detected during search mode sets the counter to 16 and begins operation. The DPLL output clock then rises at a count of 0 and falls at 16. Data is sampled on the rising edge of the clock. When a transition in the data stream is detected, the count length is adjusted by one or two counts, depending on the counter value when the transition occurs.

Count When Transition Detected	Count Length Adjustment	Counter Reset After Count Reaches
0 - 7	-2	29
8 - 15	-1	30
16 - 23	+1	32
24 - 31	+2	33
None detected	0	31

The count length adjustments cause the rising edge of the DPLL output clock to converge to the nominal center of the bit cell. In the worst case, which occurs when a DPLL pulse is coincident with the data edge, the DPLL converges after 12 data transitions.

For NRZ encoded data, a stream of alternating ones and zeros should be used as a synchronizing pattern. For NRZI encoded data, a stream of zeros should be used.

FM MODE OPERATION

FM operation is used with FM0, FM1, and Manchester data encoding. With this type of encoding, transitions in the data stream always occur at the beginning of the bit cell for FM0 and FM1, or at the center of the bit cell for Manchester. The DPLL 6-bit counter is incremented by a 32X clock. The first edge detected during search mode sets the counter to 16 and begins operation. The DPLL receiver clock then rises on a count of 8 and falls on 24. (The DPLL transmitter output clock rises on a count of 0 and falls on 16.) This provides a 1X clock with edges positioned at the nominal centers of the two halves of the bit cell. The transition detection circuit is enabled between counts of 8 and 23 inclusive. When a transition is detected, the count length is adjusted by one, depending on when the transition occurs:

Count When Transition Detected	Count Length Adjustment	Counter Reset After Count Reaches
8-15	-1	30
16-23	+1	32
24-7	Disabled	
None detected	0	31

If a transition is not detected for two consecutive data bits, the DPLL is forced into search mode and the DPLL error status bit (TRSR [3]) is asserted. This feature is disabled when the DPLL output is used only as the transmitter clock. To prevent the DPLL from locking on the wrong edges of the data stream, an opening PAD sequence should be transmitted. For FM0, a stream of at least 16 ones should be sent initially. For FM1, a minimum stream of 16 zeros should be sent and for Manchester encoding the initial data stream should consist of alternating ones and zeros.

COUNTER/TIMER

Each channel of the DUSCC contains a counter/timer (C/T) consisting of a 16-bit down counter, a 16-bit preset register, and associated control circuits. Operation of the counter/timer is programmed via the counter/timer control register (CTCR). There are also four commands associated with C/T operation, as described in the Command Description section.

Advance Information

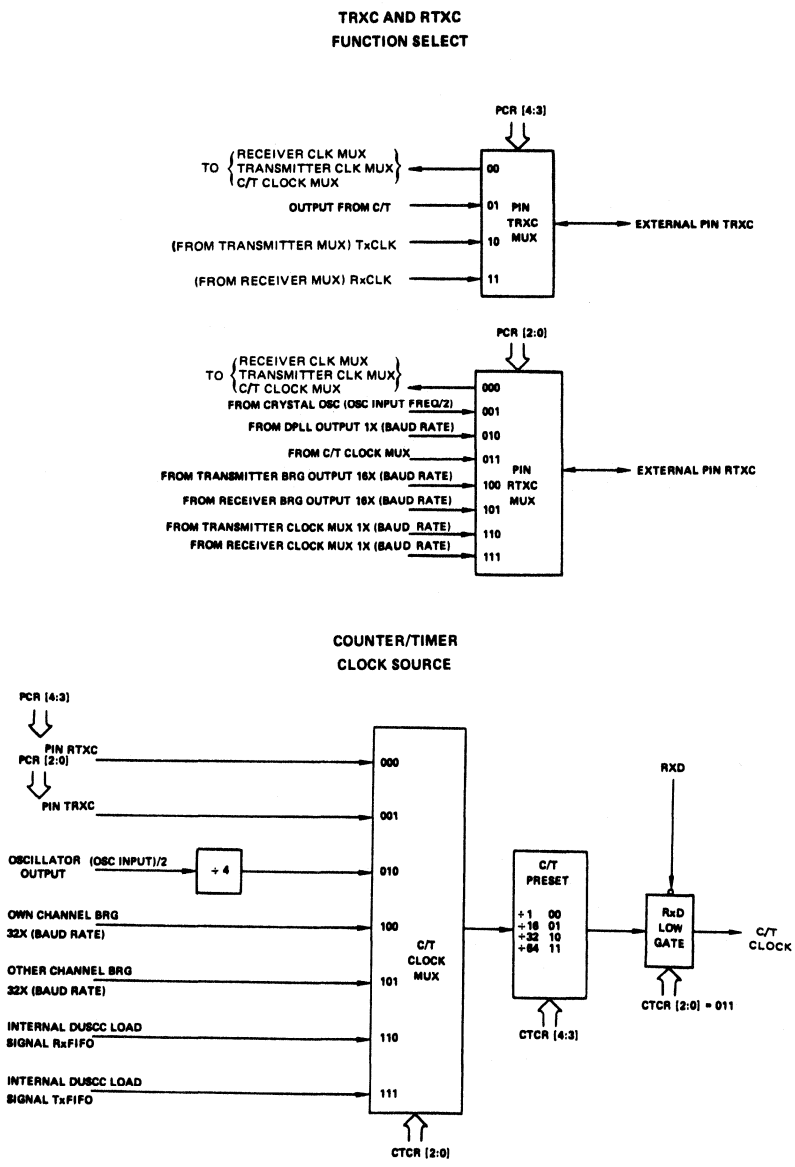


Figure 6. Timing Block (Continued)

Advance Information

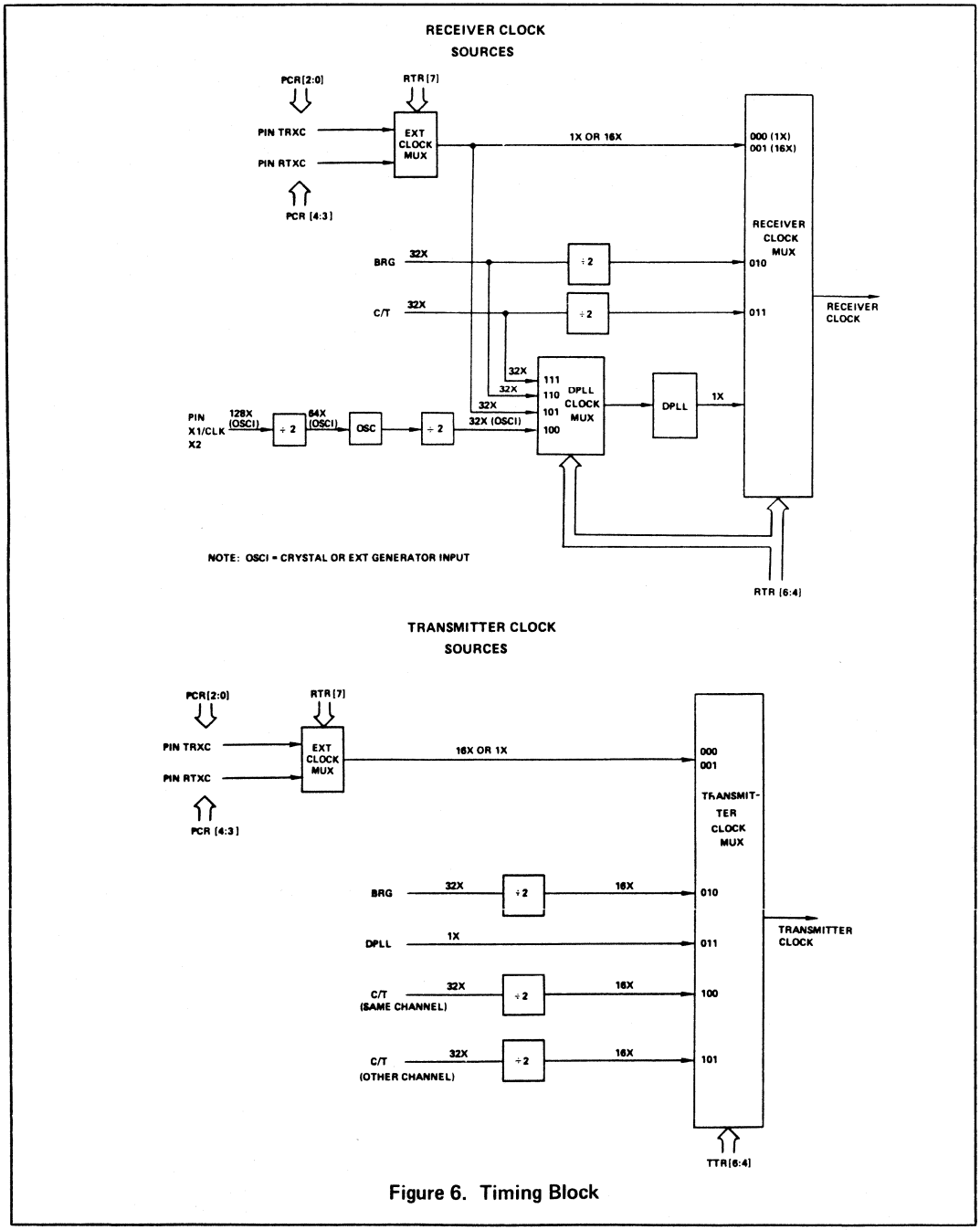


Figure 6. Timing Block

Advance Information

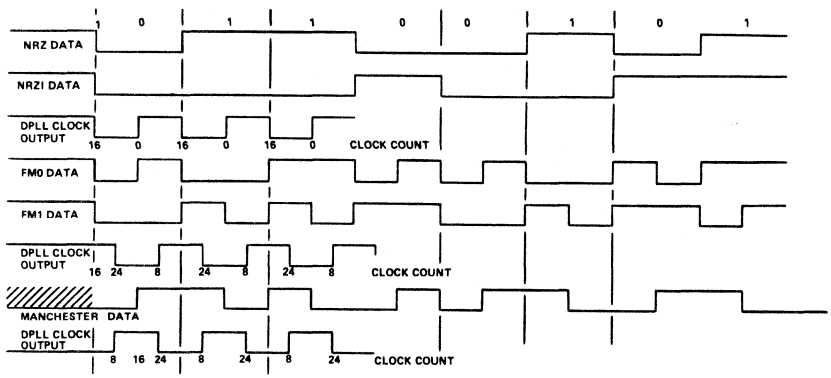


Figure 7. DPLL Waveforms

The C/T clock source, clock prescaling, and operating mode are programmed via CTCR[2:0], CTCR[4:3], and CTCR[6] respectively. The preset register is loaded by the CPU and its contents can be transferred into the down counter by a command, or automatically upon reaching terminal count if CTCR[6] is asserted. Commands are also available to stop and start the C/T and to preset it to an initial value of FFFF. The C/T zero count status bit, ICTSR[6], is asserted when the C/T reaches the terminal count of zero and ICTSR[7] indicates whether the counter is currently enabled or not.

An interrupt is generated upon reaching zero count if CTCR[7] and the channel's master interrupt enable are asserted. The output of the C/T can be programmed to be output on the channel's RTXC or TRXC pin (via PCR[4:0]) as either a single pulse or a square wave, as programmed in CTCR[5]. The contents of the C/T can be read at any time by the CPU, but the C/T should normally be stopped before this is done. Several C/T operating modes can be selected by programming of the counter/timer control register. Typical applications include:

1. Programmable divider. The selected clock source, optionally prescaled, is divided by the contents of the preset register. The counter automatically reloads itself each time the terminal count is reached.

   In this mode, the C/T may be programmed to be used as the Rx or Tx bit rate generator, as the input to the DPLL, or it may be output on a pin as either a pulse or a square wave. The C/T interrupt should be disabled in this mode.

Advance Information

2. Periodic interrupt generator. This mode is similar to the programmable divider mode, except that the C/T interrupt is enabled, resulting in a periodic interrupt to the CPU.
3. Delay timer. The counter is preset from the preset register and a clock source, optionally prescaled, is selected. An interrupt is generated upon reaching terminal count. The C/T continues counting without reloading itself and its contents may be read by the CPU to allow additional delay past the zero count to be determined.
4. Character counter. The counter is preset to FFFF by command and the clock source becomes the internal signal used to control loading of the Rx or Tx characters. This operation is selected by CTCR [2:0]. The C/T counts characters loaded into the RXFIFO by the receiver or loaded into the transmit FIFO by the CPU respectively. The current character count can be determined by the CPU by reading the contents of the C/T and taking its ones complement. Optionally, a preset number may be loaded into the counter and an interrupt generated when the count is exhausted. When counting Tx characters, the terminal count condition can be programmed through TPR [4] to cause an end of message sequence to be transmitted. When counting received characters, the FIFO'ed EOM status bit is asserted when the character which causes the count to go to zero is loaded into the receive FIFO.
5. External event counter. The counter is preset to FFFF by command and an external clock source is selected. The current count can be determined by the CPU by reading the contents of the C/T and taking its ones complement. Optionally, a preset number may be loaded into the counter and an interrupt generated when the count is exhausted.
6. Bit length measurement. The counter is preset to FFFF by command and the X1/CLK gated by RxD mode (optionally prescaled) is programmed. The C/T starts counting when RxD goes low and stops counting when RxD goes high. At this time, status and an interrupt (if enabled) are generated. The resulting count in the counter can be read by the CPU to determine the bit rate of the input data. Normally this function is used for asynchronous operation.

COMMUNICATIONS CHANNELS A AND B

Each communication channel of the DUSCC is a full duplex receiver and transmitter that supports async, COP, and BOP transmission formats. The bit rate clock for each receiver and transmitter can be selected independently to come from the bit rate generator, C/T, DPLL, or an external input (such as a modem generated clock).

Transmitter TxFIFO and TXRDY

The transmitter accepts parallel data from the data bus and loads it into the TxFIFO, which consists of four 8-bit holding registers. This data is then moved to the transmitter shift register, TxSR, which serializes the data according to the transmission format programmed. The TxSR is loaded from the TxFIFO, from special character logic, or from the CRC/LRC generator. The LSB is transmitted first, which requires right justification of characters by the CPU. TXRDY (GSR[5] or GSR[1]) and underrun (TRSR[7]) indicate the state of the TxFIFO. The TxFIFO may be addressed at any of four consecutive locations (see table 1) to allow use of multiple byte word instructions. A write to any valid address always writes data to the next empty FIFO location.

TXRDY is set when there is an empty position in the TxFIFO (OMR[4] = 0) or when the TxFIFO becomes empty (OMR[4] = 1). The CPU may reset TXRDY through a status reset write cycle. If this is done, it will not be reasserted until a character is transferred to the TxSR (OMR[4] = 0) or when the TxFIFO becomes empty again (OMR[4] = 1). The assertion of TXRDY, enabling of the IER [6] and the enabling of the channel master interrupt ICR [0] or [1] allow an interrupt to be generated.

If DMA operation is programmed, either RTXDRQ (half duplex) or TXDRQ (full duplex) follows the state of TXRDY if the transmitter is enabled. These operations differ from normal ready in that the request signal is negated on the leading edge of the DMA acknowledge signal when the subsequent transfer causes the transmit FIFO to become full, while the TXRDY signal is negated only after the transfer is completed.

Underrun set indicates that one or more data characters (not PAD characters) have been transmitted and the TxFIFO and TxSR are both empty.

In 'wait on Tx', a write to a full FIFO causes the write cycle to be extended until a FIFO position is available. DTACKN is asserted to acknowledge acceptance of the data. In non-wait modes, if an attempt is made to load data into a full TxFIFO; the TxFIFO data is preserved and the overrun data character(s) is lost. A normal DTACKN will be issued, and no indication of this occurrence is provided.

The transmitter is enabled by the enable transmitter command. When the disable transmitter command is issued, the transmitter continues to operate until the TxFIFO becomes empty.

Advance Information

TX RTS Control
If TX RTS CONTROL, TPR[3], is programmed, the channel's RTS output is negated five bit times after the last bit (stop bit in async mode) of the last character is transmitted. RTS is normally asserted and negated by writing to OMR [0]. The assertion of [3] causes RTS to be reset automatically (if the transmitter is enabled) after all characters in the transmitter FIFO (if any) are transmitted and five bit times after the "last character" is shifted out. This feature can be used to automatically terminate the transmission of a message as follows:

- Program auto-reset mode: TPR_[3] = 1.
- Enable transmitter.
- Assert RTS_N: OMR_[0] = 1.
- Send message.
- Disable transmitter after the last character is loaded into the TxFIFO.
- The last character will be transmitted and OMR_[0] will be reset five bit times after the last bit, causing RTS_N to be negated. The TxD output will remain in the marking state until the transmitter is enabled again.

The "last bit" in async is simply the last stop bit of the character. In BOP and COP, the last character is defined either explicitly by either appending it with TEOM or implicitly through the selection of the frame underrun abort sequence, TPRA[7:6] (transmitter parameter register). The table below summarizes the relationship of the selected abort sequence and the protocol mode:

TPRA [7:6]	Protocol	Last Character
00	BOP	FLAG following either FCS (if selected) or last data character
	COP	Last byte of FCS before line begins SYN or MARKing
10	BOP	Abort sequence (11111111) prior to MARKing
	COP	Last byte of FCS before line begins SYN or MARKing
11	BOP	Abort sequence (11111111) prior to FLAG
	COP	First SYN of SYN sequence

Tx CTS Operation
If CTS enable TX, TPR[2], is set, the CTS input must be asserted for the transmitter to operate. Changes in CTS while a character is being transmitted do not effect transmission of that character. However, if the CTS input becomes negated when TPR[2] is set and the transmitter is enabled and ready to start sending a character, CTS underrun, RTSR[6], is asserted and the TxD output is placed in the marking (high) state. In async mode, operation resumes when CTS is asserted again. In COP and BOP modes, the transmission of the message is terminated and operation of the transmitter will not resume until CTS is asserted and a TSOM or TSOMP command is invoked. Prior to issuing the command and retransmitting the message, the transmitter must be reset.

Special Bit Pattern Transmission
The DUSCC provides features to transmit the following special bit patterns:

Protocol	Bit Pattern
Async-Break	An all 0's character including parity bit (if specified) and stop bits. Used for send break command.
COP-SYN	Contained in S1R (single SYN mode) or in S1R/S2R (dual SYN modes). Used for send SOM and send SOM with PAD commands and for non-transparent mode linefill and IDLE.
COP-DLE	Used for send DLE command and for BISYNC transparent mode linefill and to generate BISYNC control sequences.
COP-CRC	16/8 bits from the CRC/LRC accumulator used for send EOM command or for auto-EOM modes.
BOP-FLAG	01111110. Used for send SOM, send SOM with PAD, and send EOM commands, for auto-EOM modes, and as an IDLE line fill.
BOP-ABORT	11111111. Used for send ABORT command.
BOP-CRC	16/8 bits from the CRC accumulator used for send EOM command or for auto-EOM modes.

The TxD pin is held marking after a hardware reset or a reset Tx command, when the transmitter is not enabled, if the transmitter is enabled and the TxFIFO is empty (async), or if a send SOM or SOM with PAD command has not been issued (sync modes).

The following command bits can be appended to characters in the TxFIFO: send EOM, send DLE, exclude from CRC, and reset Tx CRC. An invoked command(s) is appended to the next character loaded into the TxFIFO and follows the character through the FIFO until that character is ready to be loaded into the TxSR.

The following describes the operation of the transmitter for the various protocols.

ASYNC MODE
Serialization begins when the TxFIFO data is loaded into the TxSR. The transmitter first sends a start bit, then the programmed number of bits/character (TPR[1,0]), a parity bit (if specified), and the programmed number of stop

Advance Information

bits. Following the transmission of the stop bits, if a new character is not available in the TxFIFO, the TxD output goes to marking and the underrun condition (TRSR[7]) is set.

Transmission resumes when the CPU loads a new character into the TxFIFO or issues a send break command. The send break command clears the TxFIFO and forces a continuous space (low) on the TxD output after the character in TxSR (if any) is serialized. A send break acknowledge (TRSR[4]) is returned to the CPU to facilitate reassertion of the send break command in order to send an integral number of break characters. The send break condition is cleared when the reset Tx or disable Tx command is issued.

COP MODES

Transmitter commands associated with all COP modes are: send SOM (TSOM, send start of message), send SOM with PAD (TSOMP), send EOM (TEOM, send end of message), reset Tx CRC, exclude from CRC, and send DLE.

A send SOM or send SOM with PAD command must be issued to start COP transmission. Send SOM (without PAD) causes the Tx CRC/LRC generator to be initialized and one or two SYN characters from S1R/S2R to be loaded into the TxSR and shifted out on the TxD output. A parity bit, if specified, is appended to each SYN character after the MSB. Send SOM acknowledge (TRSR[4]) is asserted when the SYN output begins. The user may reinvoke the command to cause multiple SYNs to be transmitted. If the command is not reinvoked and the TxFIFO is empty, SYN patterns continue to be transmitted until the TxFIFO is loaded. If data is present in the FIFO, the first character is loaded into the TxSR and serialization of the data begins. Note that the TxFIFO may be preloaded with data before the send SOM is issued.

The send SOM with PAD command causes all characters in the TxFIFO (PAD characters) to be loaded into the TxSR and serialized if the Tx is enabled. Unlike the send SOM without PAD, data (non-PAD characters) cannot be preloaded into the TxFIFO. While the PAD is transmitted, parity is disabled and character length is automatically set to 8 bits. When the TxFIFO becomes empty after the PAD, the Tx CRC/LRC generator is initialized, the SYN character(s) are transmitted with optional parity appended, and send SOM acknowledge asserted. Operation then proceeds in the same manner as the send SOM command; the user has the option to invoke the send SOM command to cause multiple SYNs to be transmitted.

After the send SOM/send SOM with PAD command is executed, characters in the TxFIFO are loaded into the TxSR and shifted out with a parity bit, if specified, appended after the MSB. If the TxFIFO is empty, after the opening SYN(s) and first character have been transmitted, a data underrun condition results and TRSR[7] is asserted. The transmitter's action on data underrun is determined by TPR[7:6] and the COP protocol. If TPR [7:6] = '10', the transmitter linefills with MARK characters until a character is loaded into the FIFO. If TPR [7:6] = '11' is selected, the transmitter linefills with SYN, SYN1-SYN2, or DLE-SYN1 for mono-sync, dual sync, and BISYNC transparent modes respectively. If TPR[7:6] = '00', the BCC characters are transmitted and frame complete (TRSR[6]) is set. TxD then assumes the programmed idle state (TPR[5]) of MARKs or SYN1/SYN1-SYN2.

Operation resumes with the transmission of a SYN sequence when a TSOM command is invoked. A TSOMP command is ignored unless the transmitter is disabled and then reenabled.

An appended TEOM also terminates the frame as described above. It occurs after transmission of the character to which the TEOM is appended. The TEOM can be explicitly asserted by the send TEOM command. If TPR[4] = '1', the TEOM is automatically appended to a character in DMA mode, if the DONEN input is asserted when a character is loaded into the TxFIFO, or if the counter/timer is counting transmitted characters when the character which causes the counter to reach zero count is loaded.

The send DLE command when appended to a character in the TxFIFO, causes the DLE character to be loaded into the TxSR and serialized before the TxFIFO character is loaded into the TxSR and serialized. This feature is particularly useful for BISYNC operation. The DLE character will be excluded from the CRC accumulation in BISYNC transparent mode (see below), but will be included in all other COP modes.

In BISYNC mode, transmission of a DLE-STX character sequence (either via a send DLE command appended to the STX character, or via DLE and STX loaded into the TXFIFO) puts the transmitter into the transparent text mode of operation. In this mode, normally restricted character sequences can be transmitted as "normal" bit sequences. The switch occurs after transmission of the two characters, so that the DLE and STX are included in the BCC accumulation. If the DLE-STX is to be excluded from the CRC, the user should issue a 'reset CRC' command prior to loading the next character.

Another method of excluding the two characters from the CRC is to invoke the 'exclude from CRC'

Advance Information

command prior to loading the character(s) into the FIFO. While in transparent mode, the transmitter line fills with DLE-SYN1 and automatically transmits an extra DLE if it finds a DLE in the Tx FIFO ('DLE stuffing'). The transmitter reverts to non-transparent mode when the frame is terminated by a TEOM.

CRC/LRC accumulation can be specified in all COP modes; the type of CRC is specified via CMR2[2:0]. The send SOM/SOM with PAD commands set the CRC/LRC accumulator to its initial state and accumulation begins with the first non-SYN character after the initial SYN(s) are transmitted. PAD characters are not subject to CRC accumulation. In non-BISYNC or BISYNC normal modes, all transmitted characters except linefill characters (SYNs or MARKs) are subject to accumulation. In BISYNC transparent mode, odd (stuffed) DLEs and the DLE-SYN linefill are excluded from the accumulation. Characters can be selectively excluded from the accumulation by invoking the 'exclude from CRC' command prior to loading the character into the FIFO.

Accumulation stops when transmission of the first character of the BCC begins. The CPU can set the accumulator to its initial state prior to the transmission of any character by using the appended reset CRC command. The CRC generator is also automatically initialized after the EOM is sent.

BOP MODES

Transmitter commands associated with BOP modes are send SOM, send SOM with PAD, send EOM, and send ABORT.

The send SOM and send SOM with PAD are identical to COP modes except that a FLAG character (01111110) is used as the start of message sequence instead of the SYNs and FLAG(s) that continue to be sent until the TxFIFO is loaded. There is no zero insertion (see below) during transmission of the PAD characters.

The first characters loaded into the TxSR from the TxFIFO are the address and control fields, which have fixed character lengths of eight bits. The number of address field bytes is determined by CMR1[4:3]. If extended address field is specified, the field is terminated if the first address octet is H'00' or if the LSB of the octet is a 1. The number of control field bytes is selected by CMR1_[5]. If any information field characters follow the control field (forming an I field), they are transmitted with the number of bits per character programmed in TPR[1:0]. The TEOM can be appended to the last character either explicitly or automatically as described for COP mode. When the character with the appended TEOM is loaded from the TxFIFO, it is transmitted with the character length specified by OMR[7:5]. In this way, a residual character of 1-8 bits is transmitted without requiring the CPU to change the Tx character length for this last character.

After the opening FLAG and first address octet have been transmitted, an underrun occurs (TRSR[7] = 1) if the TxFIFO is empty when the transmitter requires a new character. The underrun control bits (TPR[7:6]) determine whether the transmitter line fills with either ABORT-MARKs, ABORT-FLAGs (see below), or ends transmission with the "normal" of the end of message sequence.

EOM on underrun is functionally similar to EOM due to an appended TEOM command. If the EOM is due to underrun, the normal character length applies to the last data character. After the last character is transmitted, the FCS (inverted CRC) and closing FLAG are sent, frame complete (TRSR[6]) is set, and the Tx CRC is initialized. If the TxFIFO is empty after the closing FLAG has been sent, TxD will assume the programmed idle state of FLAGs or MARKs (TPR[5]) and wait for a character to be loaded into the FIFO or for a TSOM command to be issued. If the TxFIFO is not empty at that time, the TxFIFO data will be loaded into the TxSR and serialized. In that case, the closing FLAG is the opening FLAG of the next frame.

The user can control the number of FLAGs between frames by invoking the send SOM command after frame complete is asserted. The DUSCC then operates in the same manner as for transmission of multiple FLAGs at the beginning of a frame. When the command is no longer reinvoked, transmission of the TxFIFO data will begin. If the FIFO is empty, FLAGs continue to be transmitted.

The DUSCC provides automatic zero insertion in the data stream to prevent erroneous transmission of the FLAG sequence. All data characters loaded into the TxSR from the TxFIFO and characters transmitted from the CRC generator are subject to zero insertion. For this feature, a zero is inserted in the serial data stream each time five consecutive ones (regardless of character boundaries) have been transmitted.

A send ABORT command clears the TxFIFO and inserts an ABORT character of eight ones (not subject to zero insertion) into the TxSR for transmission after the current character has been serialized. A send abort ack (TRSR[4]) facilitates reassertion of send abort by the user to guarantee transmission of multiple abort characters. This feature can be used to send the

Advance Information

15-ones idle sequence. The transmitter sends either marks or FLAGs after the abort character(s) has been transmitted, depending on TPR[7:6]. Operation resumes with the transmission of a FLAG when a TSOM command is invoked. A TSOMP command is ignored unless the transmitter is disabled and then re-enabled.

CRC accumulation can be specified in all BOP modes. The type of CRC is specified via CMR2[2:0], and is normally selected as CRC-CCITT preset to ones, although any option is valid.

The send SOM/SOM with PAD command sets the CRC accumulator to its initial state and accumulation begins with the first address octet after the initial FLAG(s). Accumulation stops when transmission of the first character of the FCS begins. The CPU can set the accumulator to its initial state prior to the transmission of any character by using the appended reset CRC command and can exclude any character from the accumulation by use of the exclude from CRC command, but these features would not normally be used in BOP modes. The CRC generator is also automatically initialized after the EOM or an ABORT are sent.

Receiver RxFIFO, RXRDY

The receiver converts received serial data on RxD (LSB first) into parallel data according to the transmission format programmed. Data is shifted through a synchronizing flip flop and one or more shift registers, the last of which is the 8-bit receiver shift register (RxSR). Bits are shifted into the RxSR on the rising edge of each 1X receive clock until the LSB is in RxSR[0]. Hence, the received character is right justified, with all unused bits in the RxSR cleared to zero. A receive character length counter generates a character boundary signal for synchronization of character assembly, character comparisons, break detection (async), and RxSR to RxFIFO transfers (except for BOP residual characters). During COP and BOP hunt phases, the SYN/FLAG comparison is made each receive bit time, as are flag, abort, and idle comparisons in BOP modes.

An internal clock, from the BRG or the counter/timer, or an external 1x or 16x clock may be used as the receiver clock in async mode. The BRG or counter/timer cannot be used for the receiver clock in synchronous modes, since these modes require a 1X receive clock that is in phase with the received data. This clock may come externally from the RTXC or TRXC pins, or it may be derived internally by the DPLL. Encoded data is converted to NRZ format for the receiver circuits by using clock pulses generated by the DPLL.

When a complete character has been assembled in the RxSR, it is loaded into the receive FIFO with appended status bits. The most significant data bits of the character are set to zero if the character length is less than eight bits. In async and COP modes the user may select, via RPR[3], whether the data transferred to the FIFO includes the received parity bit or not. The receiver indicates to the CPU or DMA controller that it has data in the FIFO by asserting the channel's RXRDY status bit (GSR[4] or GSR[0]) and, if in DMA mode, the corresponding receiver DMA request pin.

The RxFIFO consists of four 8-bit holding registers with appended status bits for character compare indication (async), EOM indication (BISYNC/BOP), and parity, framing, and CRC errors. Data is loaded into the RxFIFO from the RxSR and extracted (read) by the CPU or DMA controller via the data bus. An RXFIFO read creates an empty RxFIFO position for new data from the RxSR.

RXRDY is asserted as follows, depending on the state of OMRB[3]:

1. If OMRB[3] is 0 (FIFO not empty), RXRDY is asserted each time a character is transferred from the receive shift register to the receive FIFO. If it is not reset by the CPU, RXRDY remains asserted until the receive FIFO becomes empty, at which time it is automatically negated. If it is reset by the CPU, it will remain negated, regardless of the current state of the receive FIFO, until a new character is transferred from the RXSR to the RXFIFO.
2. If OMRB[3] is 1 (FIFO full), RXRDY is asserted:
   a. when a character transfer from the receive shift register to the receive FIFO causes it to become full
   b. when a character with a tagged EOM status bit is loaded into the FIFO (BISYNC or BOP)
   c. when the counter/timer is programmed to count received characters and the character which causes it to reach zero count is loaded into the FIFO (ICTSR [6])
   d. when the beginning of a break is detected in async mode.

If it is not reset by the CPU, RXRDY remains asserted until the FIFO becomes empty, at which time it is automatically negated. If it is reset by the CPU, it will remain negated regardless of the current state of the receive FIFO, until it is asserted again due to one of the above conditions.

Advance Information

The assertion of RXRDY_ causes an interrupt to be generated if IER_[4] and the channel's master int enable (ICR[0] or ICR[1]) are asserted.

When DMA operation is programmed, the RXRDY_ status bit is routed to the DMA control circuitry for use as the channel receiver DMA request. Assertion of RXRDY_ results in assertion of RXDREQ_ .

Several status bits are appended to each character in the RxFIFO. When the FIFO is read, causing it to be 'popped', the status bits associated with the new character at the top of the RxFIFO are logically OR'ed into the RSR. Therefore, the user should read RSR before reading the RxFIFO in response to RxRDY activation. If character-by-character status is desired, the RSR should be read and cleared each time a new character is received. The user may elect to accumulate status over several characters or over a frame by clearing RSR at appropriate times. This mode would normally also be used when operating in DMA mode. If the RxFIFO is empty when a read is attempted, and wait mode is not being used, a H'00' is output on the data bus.

In all modes, the DUSCC protects the contents of the FIFO and the RxSR from overrun. If a character is received while the FIFO is full and a character is already in the RxSR waiting to be transferred into the FIFO, the overrunning character is discarded and the OVERRUN status bit (RSR[5]) is asserted. If the overrunning character is an end-of-message character, the character is lost but the FIFO'ed EOM status bit will be asserted when the character in the RxSR is loaded into the FIFO.

Operation of the receiver is controlled by the enable receiver command. When this command is issued, the DUSCC goes into the search for start bit state (async), search for SYN state (COP modes), or search for FLAG state (BOP modes). When the disable receiver command is issued, the receiver ceases operation immediately. The RxFIFO is cleared on RESET, by a reset receiver command, or on receipt of a break (async) or abort (BOP). However, disabling the receiver does not affect the RxFIFO, RxRDY, or DMA request operation.

If DCD enable RX, RPR[2], is asserted, the DCD input must be asserted for the receiver to operate. The DCD input is sampled approximately every 6.8usec. If RPR[2] is asserted and the sampling circuit detects that the DCD input has been negated, the receiver ceases operation. Operation resumes when the sampled DCD is asserted again. A change of state detector is provided on the DCD input of each channel. The user may program a change of state to cause an interrupt to be generated (master interrupt enable ICR[0] or [1] and IER_ [7] must be set) so that appropriate action can be taken.

If RX RTS control, RPR[4], is asserted, the channel's RTS output is negated automatically when the start bit of a new character is received with the FIFO full. This may be used as a flow control signal to prevent overrun, by connecting the RTS output to the CTS input of the transmitter. The new character will be assembled in the RxSR, but its transfer to the FIFO will be delayed until the CPU reads the FIFO, making a FIFO position available for the new character. The negation of RTS is indicated by RSR[6].

In async mode, RSR [6] controls the deactivation of the RTS_N output by the receiver. RTS can be manually asserted and negated by writing to OMR [0]. The assertion of [4] causes RTS to be reset automatically upon receipt of a valid start bit if the channel's receive FIFO is full. When this occurs, the RTS negated status bit (RSR[6]) is set. This feature can be used to prevent overrun in the receiver by using the RTS_N output signal to control the CTS_N input of the remote transmitter.

Once enabled, receiver operation depends on channel protocol mode. The following describes the receiver operation for the various protocols:

ASYNC MODE

When first enabled, the receiver goes into the search for start bit state, looking for a high to low (mark to space) transition of the start bit on the RXD input. If a transition is detected, the state of the RXD pin is sampled again each 16X clock for 7-1/2 clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RXD is sampled high, the start bit is invalid and the search for a valid start bit begins again.

If RXD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals (16 periods of the 16X Rx clock; one period of the 1X Rx clock) at the theoretical center of the bit, until the proper number of data bits and the parity bit (if specified) have been assembled, and the first stop bit has been sampled.

The assembled character is then transferred to the RxFIFO with appended parity error (if parity is specified) and framing error status bits. The DUSCC can be programmed to compare this character to the contents of S1R. The appended character compare status bit is set if the data matches and there is no parity error.

Advance Information

After the stop bit is sampled, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (i.e. framing error) and RXD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one half bit time after the stop bit was sampled).

If a break condition is detected (RXD low for entire character time including optional parity and first stop bit), only one character consisting of all zeros will be loaded into the RxFIFO and break start detect, RSR[2], will be set (and also framing error). The RXD input must return to a high condition for one half of a bit time (16X clock mode) or for two successive clock edges (1X clock mode) before the break condition is terminated and the search for the next start bit begins. At that time, the break end detect condition, RSR[3], is set.

COP Modes

When the receiver is enabled in COP modes, it first goes into the SYN hunt phase, testing the received data each bit time for receipt of the appropriate SYN bit pattern, plus parity if specified, to establish character boundaries. Receipt of the SYN bit pattern terminates hunt phase and places the receiver in the data phase. In COP single SYN protocol mode, S1R contains the SYN character required to establish character synchronization. In COP dual SYN and BISYNC protocol modes, S1R and S2R contain the first and second SYN characters required to establish character synchronization. The SYN character length is the same as the character length programmed in RPR[1:0], and does not include the parity bit if parity is specified. SYN characters received with a parity error, when parity is specified, are considered invalid and will not cause synchronization to be achieved.

If external synchronization is programmed (RPR[4] = 1), the internal SYN detection and special character recognition logic are disabled and receipt of SYN characters is not required. A pulse on the SYNIN input pin will establish character synchronization and terminate hunt phase. The SYNIN pin is ignored after the first input on the SYNIN pin is received. The receiver must be disabled and then reenabled to resynchronize or to return to normal mode.

The SYN detect status bit (RSR[2]), is set whenever SYN1, SYN1-SYN2, or DLE-SYN1 is detected for single SYN, dual SYN/BISYNC normal, and BISYNC transparent modes, respectively.

The SYNOUT pin (48 pin version) will go active for one receive clock period after SYN detection. After character sync has been attained, the receiver enters the data phase, the receiver assembles characters in the RxSR with the least significant bit received first. It computes the BCC if specified, checks parity if specified, and checks for overrun errors.

The operation of the BCC (CRC/LRC) logic depends on the particular COP mode in use. The BCC is initialized upon first entering the data phase, and may also be initialized by the user at any time by invoking the 'reset Rx CRC' command. For non-BISYNC modes, all received characters after entering data phase are included in the BCC computation, except for leading SYNs and SYNs which are specified to be stripped by RPR[7]. As each received character is transferred from the RxSR to the FIFO, the current value of the BCC characters is checked and the CRC ERROR status bit (RSR[1]) is set if the value of the CRC remainder is not the expected value. The receiver computes the BCC for text messages automatically when operating in BISYNC protocol mode.

BISYNC Features

The DUSCC provides support for both BISYNC normal and transparent operation, the following summarizes the features provided. Both EBCDIC and ASCII text messages can be handled by the DUSCC as selected by CMR1 [4]. The receiver has the capability of recognizing special characters for the BISYNC protocol mode.

BISYNC - Single character sequences:

Sequence	ASCII	EBCDIC	Description
SOH	H'81	H'01	Start of header
STX	H'82	H'02	Start of text
ETX	H'83	H'03	End of text
EOT	H'84	H'37	End of transmission
ENQ	H'85	H'2D	Enquiry
DLE	H'90	H'10	Data link escape

Sequence	ASCII	EBCDIC	Description
NAK	H'95	H'3D	Negative ack
ETB	H'97	H'26	End of block
ITB	H'9F	H'1F	Intermediate block

BISYNC - Two character sequences:

Sequence	ASCII	EBCDIC	Description
ACK0	H'90, B0	H'10, 70	Acknowledge 0
ACK1	H'90, B1	H'10, 61	Acknowledge 1
WACK	H'90, BB	H'10, 6B	Wait Before Xmit Ack
RVI	H'90, BC	H'10, 7C	Reverse Interrupt
TTD	H'82, 85	H'02, 2D	Temporary Text Delay

Advance Information

BISYNC - (Transparent text mode) - Two character sequences:

Sequence	ASCII	EBCDIC	Description
DLE-ENQ	H'90, 85	H'10, 2D	Enquiry
DLE-ITB	H'90, 9F	H'10, 1F	Intermediate Block
DLE-ETB	H'90, 97	H'10, 26	End of Block
DLE-ETX	H'90, 83	H'10, 03	End of Text
DLE-STX	H'90, 82	H'10, 02	Transparent text mode.

All the above sequences with the exception of SOH and STX when detected explicitly cause a status to be affected. The following describes the conditions when this occurs.

The first character received when entering data phase for a text message should be an SOH, an STX, or a DLE-STX two character sequence. Receipt of any of these initializes the CRC generator and starts the CRC accumulation. The SOH places the receiver in header mode, receipt of the STX places it in text mode, and receipt of the DLE-STX sequence (at any time) automatically places the receiver in transparent mode and sets the XPNT mode status bit, TRSR[0]. There is no explicit status associated with SOH and STX. If any other characters are received when entering the data phase, the message is not treated as a text message.

After the data phase is established, the receiver searches the data stream for an end of message control character:

Header field: ENQ, ETB, or ITB
Normal text field: ENQ, ETX, ETB, or ITB
Transparent text field: DLE-ENQ, DLE-ETX, DLE-ETB, or DLE-ITB
Not in header or text: EOT, NAK, ACK0, ACK1, WACK, RVI or TTD.

Detection of any one of these termination sequences causes RSR[7] to be set. Also if the receiver does not detect a closing PAD (four 1's) after the 'EOT' or 'NAK', the PAD ERROR RSR [7] is set. When the abort sequence ENQ or DLE-ENQ is detected, the character is tagged with an REOM status and transferred to the FIFO, but the appended BCC error status bit should be ignored. For the other EOM control sequences, the receiver waits for the next two bytes (the CRC bytes) to be received, checks the value of the CRC generator, and tags the transferred character with a CRC ERROR (RSR[1]) if the CRC remainder is not correct.

The CRC bytes are normally not transferred to the FIFO, unless the transfer FCS to FIFO control bit (RPR[6]) is asserted. In this case the EOM and CRC error status bits will be tagged onto the last byte of the last BCC byte instead of to the last character of the message. After detecting one of the end-of-message (EOM) character sequences and setting PSR [7], the receiver goes into auto hunt for the next character sync and PAD check if PRP [5] is set.

SYN Pattern Stripping

The first one (single SYN mode) or two (dual SYN and BISYNC modes) detected SYN character(s) are always stripped and excluded from the BCC, but SYN patterns after character synchronization are treated by the receiver according to the RPR [7]. SYN character patterns are defined for the various COP modes as follows:

COP single SYN mode - SYN1

COP dual SYN mode - SYN1 and SYN2 when immediately preceded by SYN1.

BISYNC normal mode - SYN1 and SYN2 when immediately preceded by SYN1. SYN1 is always stripped, even if it is not followed by SYN2 when stripping is selected.

BISYNC transparent - DLE-SYN1, where the DLE is the last of an odd number of consecutive DLEs.

0 Strip only the SYN required to establish character sync.
1 Strip all SYNs. Additionaly, strip odd DLEs when operating in BISYNC transparent mode.

Processing of the SYN patterns is determined by the RPR[7] bit, the COP mode, and the position of the pattern in the frame. This is summarized below:

Mode	RPRA[7]	Pre-Sync	After Synchronization
BISYNC	0	no FCS	no FCS
		no FIFO	pattern into FIFO
	1	no FCS	no FCS
		no FIFO	no FIFO
COP	0	no FCS	no FCS
		no FIFO	pattern into FIFO
	1	no FCS	no FCS
		no FIFO	no FIFO

The value of this field does not affect the setting of the SYN DETECT status bit and the generation of a SYNOUT pulse when a SYN pattern is received.

BOP Mode

In BOP protocol mode, the receiver may be in any one of four phases: hunt phase, address field (A) phase, control field (C) phase, or information field (I) phase. The character length for the A and C phases is always 8 bits. The I field character length is specified in RPR[1:0].

Advance Information

After an enable receiver command is executed, the receiver enters hunt phase, in which a comparison for the string (01111110) is done every Rx bit time. The FLAG delineates the beginning (and end) of a received frame and establishes the character boundary. Each FLAG match causes the FLAG detect status bit (RSR[2]) to be set and SYNOUT (48-pin DUSCC) to be activated for one receive clock period. FLAGs with an overlapping zero will be detected. All FLAGs are deleted from the data stream.

Once a FLAG has been detected, the receiver will exit hunt phase and enter address phase. The length and handling of the address field are determined by the values programmed in CMR1[2:0], which selects one of the BOP secondary address modes. The secondary address modes are selected by CMR1 [4:3] and function as described below:

Single Address Octet
For receive, the address comparison for a secondary station is made on the first octet following the opening FLAG. A match occurs if the first octet after the FLAG matches the contents of S1R, or if all parties address (RPR[3]) is asserted, and the first octet is equal to H'FF'.

Dual Address Octet
For receive, the address comparison for a secondary station is made on the first two octets following the opening FLAG. A match occurs if the first two octets after the FLAG match the contents of S1R and S2R respectively, or if all parties address (RPR[3]) is asserted if the first two octets are equal to H'FF, FF'.

Dual Address with Group Mode
For receive, the address comparison for a secondary station is made on the first two octets following the opening FLAG. A match occurs for one of three possible conditions. If the first two octets after the FLAG match the contents of S1R and S2R respectively, or if the first octet is H'FF' and the second matches the contents of S2R (group mode), or when all parties address (RPR[3]) is asserted if the first two octets are equal to H'FF, FF'. The second condition (group mode) allows a selected group of stations to receive a message.

Extended Address Mode
Extend address field to the next octet if the LSB of the current address octet is zero. Address field is terminated if the LSB of the address is a one. The address field will be terminated after the first octet if the null address H'00' is received/transmitted as the first address octet. For this mode the receiver does not perform an address comparison (all received characters after the opening FLAG are transferred to the FIFO) but does determine when the address field is terminated.

The length of the A field may be a single octet, a dual octet, or more octets, as described above. A primary station or an extended address secondary station does not perform an address comparison, and all characters in the A, C, and I fields after the flag are transferred to the FIFO. For the other secondary address modes, if there is a match, or the received character(s) match either of the other enabling conditions (group or all-parties address), all characters in the A, C, and I fields are transferred to the FIFO. If there is no match, the receiver returns to the FLAG hunt phase.

C phase begins after A phase is terminated. The receiver receives one or two control characters (CMR[5]). After this phase is terminated, the character length is switched automatically from 8 bits to the number of bits specified in RPR[1:0] and the information field phase is entered.

The frame is terminated when a closing FLAG is detected. The same FLAG can also serve as the opening FLAG of the next frame. The 16 bits received prior to the closing FLAG form the frame check sequence (if an FCS is specified in CMR2[2:0]). All non-FLAG characters of the frame are accumulated in the CRC checker and the result is compared to the expected remainder. Failure to match will cause a CRC error. EOM detect (RSR[7]), RCL not zero (RSR[0]), and CRC error (RSR[1]) are normally FIFO'ed with the last character of the I field. RCL not zero (RSR[0]) is set if the length of this character of the I field does not have the length programmed in RPR[1:0]. The residual character length in TRSR[2:0] is also valid at that time. The CRC characters themselves are normally not passed to the RxFIFO. However, if the transfer FCS to FIFO control bit (RPR[6]) is asserted, the FCS bytes will be transferred to the FIFO. In this case the EOM and CRC error status bits will be tagged onto the last byte of the last CRC byte instead of to the last character of the message.

If the closing FLAG is received prior to receipt of the appropriate number of A field, C field, and FCS field octets, a short frame will be detected and RSR[4] will be set. The I field need not be present in a valid frame. An abort (01111111) comparison is done after an opening FLAG has been received and up to receipt of the closing FLAG. A match causes the abort detect status bit (RSR[6]) to be set. The receiver then enters FLAG search. The abort is stripped from the received data stream.

Advance Information

If a zero followed by 15 contiguous ones is detected, the idle detect status bit (RSR[3]) is set. This comparison is done whenever the receiver is enabled. Therefore, it can occur before or after a received frame.

Zero deletion is performed during BOP receive. A zero after 5 contiguous ones is deleted from the data stream. Deleted zeroes are not subject to CRC accumulation. FLAG, ABORT, and IDLE comparisons are done prior to zero deletion.

If external synchronization is programmed (RPR[4] = 1), the internal FLAG detection and address comparison logic is disabled and receipt of FLAGs is not required. In this arrangement, a pulse on the SYNIN input pin will establish synchronization and terminate hunt phase. The receiver will then go immediately into the I-field mode, assembling and transferring characters into the FIFO with the character length specified in RPR[1:0]. The SYNIN pin is ignored after the first input on the SYNIN pin is received. The receiver must be disabled and then reenabled to resynchronize or to return to normal operating mode.

SUMMARY OF COP FEATURES

COP Dual Octet Address:	
SYN detect	SYN1-SYN2
Line fill	SYN1-SYN2
SYN stripping	SYN1-SYN2 used to establish character sync. Subsequent to this, SYN1 and SYN1-SYN2 if stripping is specified by RPR[7].
Excluded from FCS	SYN1 and SYN1-SYN2 at beginning of message and, if SYN stripping is specified by RPR[7], anywhere else in the message for the Rx; line fill SYN1-SYN2 for Tx.

BISYNC normal mode:	
SYN detect	SYN1-SYN2
Line fill	SYN1-SYN2
SYN stripping	SYN1-SYN2 used to establish character sync. Subsequent to this, SYN1 and SYN1-SYN2 if stripping is specified by RPR[7].
Excluded from FCS	Same as COP dual SYN mode plus additional characters as required by the protocol.

BISYNC transparent mode:	
SYN detect	*DLE-SYN1
Line fill	DLE-SYN1
SYN/DLE stripping	*DLE-SYN1 and odd DLEs if stripping is specified by RPR[7].
Excluded from FCS	*DLE-SYN1 and odd DLEs if stripping is specified by RPR[7].

* DLE indicates last DLE of an odd number of consecutive DLEs.

COP Dual Octet Address:	
SYN detect	SYN1
Line fill	SYN1
SYN stripping	SYN1 used to establish character sync. Subsequent to this, SYN1 if stripping is specified by RPR[7].
Excluded from FCS	SYN1 at beginning of message and, if SYN stripping is specified by RPR[7], anywhere else in the message for the Rx; line fill SYN1 for Tx.

Multi-Protocol Communications Controller (MPCC)

Product Specification

Originally published by Signetics February 1985

DESCRIPTION

The SCN2652/68652 Multi-Protocol Communications Controller (MPCC) is a monolithic n-channel MOS LSI circuit that formats, transmits and receives synchronous serial data while supporting bit-oriented or byte control protocols. The chip is TTL compatible, operates from a single +5V supply, and can interface to a processor with an 8 or 16-bit bidirectional data bus.

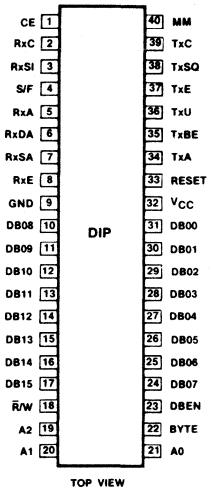
FEATURES

- **DC to 1Mbps or 2Mbps data rate**
- **Bit-oriented protocols (BOP): SDLC, ADCCP, HDLC**
- **Byte-control protocols (BCP): DDCMP, BISYNC (external CRC)**
- **Programmable operation**
  - **8 or 16-bit tri-state data bus**
  - **Error control – CRC or VRC or none**
  - **Character length – 1 to 8 bits for BOP or 5 to 8 bits for BCP**
  - **SYNC or secondary station address comparison for BCP-BOP**
  - **Idle transmission of SYNC/FLAG or MARK for BCP-BOP**
- **Automatic detection and generation of special BOP control sequences, i.e., FLAG, ABORT, GA**
- **Zero insertion and deletion for BOP**
- **Short character detection for last BOP data character**
- **SYNC generation, detection, and stripping for BCP**
- **Maintenance mode for self-testing**
- **TTL compatible**
- **Single +5V supply**

APPLICATIONS

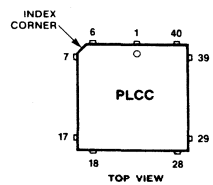
- **Intelligent terminals**
- **Line controllers**
- **Network processors**
- **Front end communications**
- **Remote data concentrators**
- **Communication test equipment**
- **Computer to computer links**

PIN CONFIGURATION



TOP VIEW

CD00160S



TOP VIEW

CD0044PS

Pin	Function	Pin	Function
1	NC	23	NC
2	CE	24	A0
3	RxC	25	BYTE
4	RxSI	26	DBEN
5	S/F	27	DB07
6	RxA	28	DB06
7	RxDA	29	DB05
8	RxSA	30	DB04
9	RxE	31	DB03
10	GND	32	DB02
12	NC	33	DB01
11	DB08	34	NC
13	DB09	35	DB00
14	DB10	36	V_{CC}
15	DB11	37	RESET
16	DB12	38	TxA
17	DB13	39	TxBE
18	DB14	40	TxU
19	DB15	41	TxE
20	R/W	42	TxSQ
21	A2	43	TxC
22	A1	44	MM

NOTE: 00 is least significant bit, highest number (that is, DB15, A2) is most significant bit.

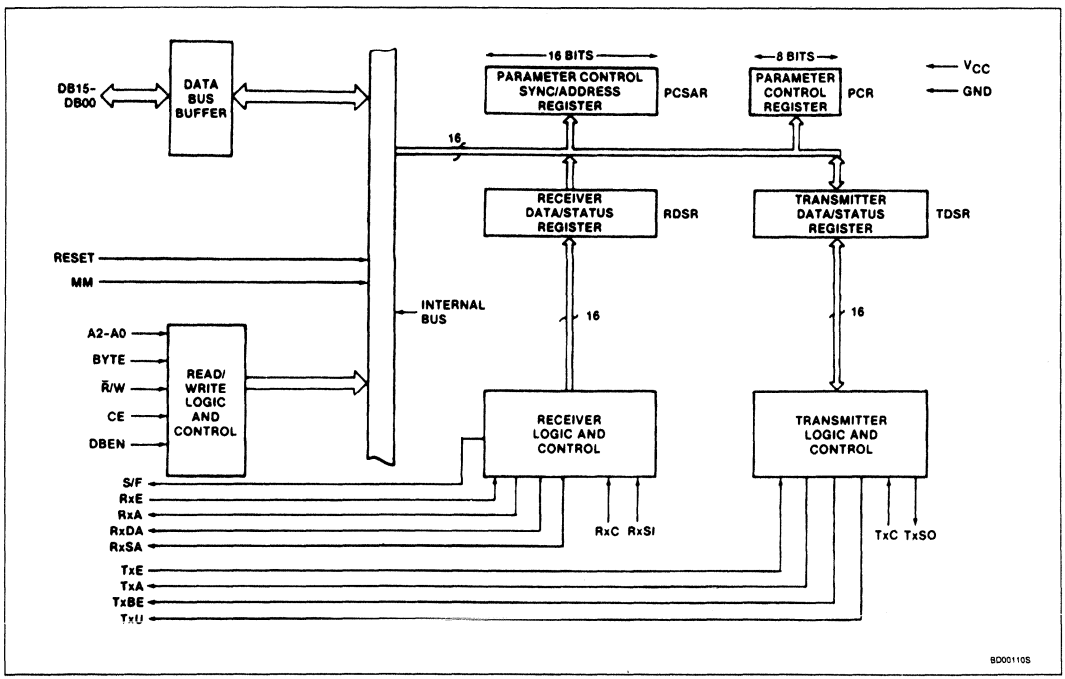
Product Specification

ORDERING CODE

PACKAGES		V_{CC} = 5V±5%		
		Commercial	Automotive	Extended
		0°C to +70°C	−40°C to +85°C	−55°C to +125°C
Ceramic DIP	1MHz	SCN2652AC1I40	SCN2652AA1I40	SCN2652AM1I40
	2MHz	SCN2652AC2I40	SCN2652AA2I40	SCN2652AM2I40
Plastic DIP	1MHz	SCN2652AC1N40	Contact Factory	Not Available
	2MHz	SCN2652AC2N40	Contact Factory	Not Available
Plastic LCC	1MHz	SCN2652AC1A44	Contact Factory	Not Available
	2MHz	SCN2652AC2A44	Contact Factory	Not Available

NOTE:
SCN68652 is identical to SCN2652. Order using part numbers shown above.

BLOCK DIAGRAM



Product Specification

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
DB15 – DB00	17 – 10 24 – 31	I/O	**Data Bus:** DB07 – DB00 contain bidirectional data while DB15 – DB08 contain control and status information to or from the processor. Corresponding bits of the high and low order bytes can be wire OR'ed onto an 8-bit bus. The data bus is floating if either CE or DBEN are low.
A2 – A0	19 – 21	I	**Address Bus:** A2 – A0 select internal registers. The four 16-bit registers can be addressed on a word or byte basis. See Register Address section.
BYTE	22	I	**Byte:** Single byte (8-bit) data bus transfers are specified when this input is high. A low level specifies 16-bit data bus transfers.
CE	1	I	**Chip Enable:** A high input permits a data bus operation when DBEN is activated.
$\overline{R}$/W	18	I	**Read/Write:** $\overline{R}$/W controls the direction of data bus transfer. When high, the data is to be loaded into the addressed register. A low input causes the contents of the addressed register to be presented on the data bus.
DBEN	23	I	**Data Bus Enable:** After A2 – A0, CE, BYTE and $\overline{R}$/W are set up, DBEN may be strobed. During a read, the 3-state data bus (DB) is enabled with information for the processor. During a write, the stable data is loaded into the addressed register and TxBE will be reset if TDSR was addressed.
RESET	33	I	**Reset:** A high level initializes all internal registers (to zero) and timing.
MM	40	I	**Maintenance Mode:** MM internally gates TxSO back to RxSI and TxC to RxC for off line diagnostic purposes. The RxC and RxSI inputs are disabled and TxSO is high when MM is asserted.
RxE	8	I	**Receiver Enable:** A high level input permits the processing of RxSI data. A low level disables the receiver logic and initializes all receiver registers and timing.
RxA	5	O	**Receiver Active:** RxA is asserted when the first data character of a message is ready for the processor. In the BOP mode this character is the address. The received address must match the secondary station address if the MPCC is a secondary station. In BCP mode, if strip-SYNC ($PCSAR_{13}$) is set, the first non-SYNC character is the first data character; if strip-SYNC is zero, the character following the second SYNC is the first data character. In the BOP mode, the closing FLAG resets RxA. In the BCP mode, RxA is reset by a low level at RxE.
RxDA*	6	O	**Receiver Data Available:** RxDA is asserted when an assembled character is in $RDSR_L$ and is ready to be presented to the processor. This output is reset when $RDSR_L$ is read.
RxC	2	I	**Receiver Clock:** RxC (1X) provides timing for the receiver logic. The positive going edge shifts serial data into the RxSR from RxSI.
S/F	4	O	**SYNC/FLAG:** S/F is asserted for one RxC clock time when a SYNC or FLAG character is detected.
RxSA*	7	O	**Receiver Status Available:** RxSA is asserted when there is a zero to one transition of any bit in $RDSR_H$ except for RSOM. It is cleared when $RDSR_H$ is read.
RxSI	3	I	**Receiver Serial Input:** RxSI is the received serial data. Mark = '1', space = '0'.
TxE	37	I	**Transmitter Enable:** A high level input enables the transmitter data path between $TDSR_L$ and TxSO. At the end of a message, a low level input causes TxSO = 1(mark) and TxA = 0 after the closing FLAG (BOP) or last character (BCP) is output on TxSO.
TxA	34	O	**Transmitter Active:** TxA is asserted after TSOM ($TDSR_8$) is set and TxE is raised. This output will reset when TxE is low and the closing FLAG (BOP) or last character (BCP) has been output on TxSO.
TxBE*	35	O	**Transmitter Buffer Empty:** TxBE is asserted when the TDSR is ready to be loaded with new control information or data. The processor should respond by loading the TDSR which resets TxBE.
TxU*	36	O	**Transmitter Underrun:** TxU is asserted during a transmit sequence when the service of TxBE has been delayed for one character time. This indicates the processor is not keeping up with the transmitter. Line fill depends on $PCSAR_{11}$. TxU is reset by RESET or setting of TSOM ($TDSR_8$), synchronized by the falling edge of TxC.
TxC	39	I	**Transmitter Clock:** TxC (1X) provides timing for the transmitter logic. The positive going edge shifts data out of the TxSR to TxSO.
TxSO	38	O	**Transmitter Serial Output:** TxSO is the transmitted serial data. Mark = '1', space = '0'.
V_{CC}	32	I	**+5V:** Power supply.
GND	9	I	**Ground:** 0V reference ground.

*Indicates possible interrupt signal

Product Specification

Table 1. GLOSSARY

REGISTERS		NO. OF BITS	DESCRIPTION*
Addressable			
PCSAR	Parameter control sync/ address register	16	$PCSAR_H$ and PCR contain parameters common to the receiver and transmitter. $PCSAR_L$ contains a programmable SYNC character (BCP) or secondary station address (BOP).
PCR	Parameter control register	8	
RDSR	Receive data/status register	16	$RDSR_H$ contains receiver status information. $RDSR_L$ = RxDB contains the received assembled character.
TDSR	Transmit data/status register	16	$TDSR_H$ contains transmitter command and status information. TDSRL = TxDB contains the character to be transmitted.
Internal			
CCSR	Control character shift register	8	These registers are used for character assembly (CSSR, HSR, RxSR), disassembly (TxSR), and CRC accumulation/generation (RxCRC, TxCRC).
HSR	Holding shift register	16	
RxSR	Receiver shift register	8	
TxSR	Transmitter shift register	8	
RxCRC	Receiver CRC accumulation register	16	
TxCRC	Transmitter CRC generation register	16	

NOTES:
*H = High byte - bits 15-8
L = Low byte - bits 7-0

FUNCTIONAL DESCRIPTION

The MPCC can be functionally partitioned into receiver logic, transmitter logic, registers that can be read or loaded by the processor, and data bus control circuitry. The register bit formats are shown in figure 1 while the receiver and transmitter data paths are depicted in figures 2 and 3.

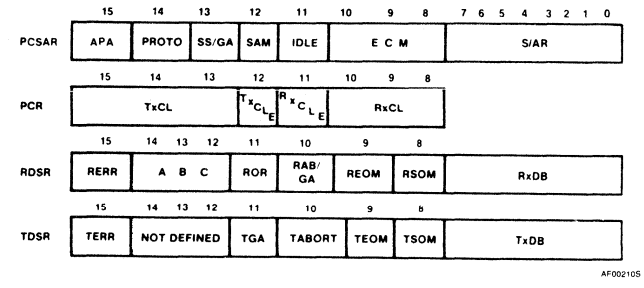
Table 2. ERROR CONTROL

CHARACTER	DESCRIPTION
FCS	Frame check sequence is transmitted/received as 16 bits following the last data character of a BOP message. The divisor is usually CRC-CCITT ($X^{16} + X^{12} + X^5 + 1$) with dividend preset to 1's but can be other wise determined by ECM. The inverted remainder is transmitter as the FCS.
BCC	Block check character is transmitted/received as two successive characters following the last data character of a BCP message. The polynomial is CRC-16 ($X^{16}+X^{15}+X^2+1$) or CRC-CCITT with dividend preset to 0's (as specified by ECM). The true remainder is transmitted as the BCC.

Table 3. SPECIAL CHARACTERS

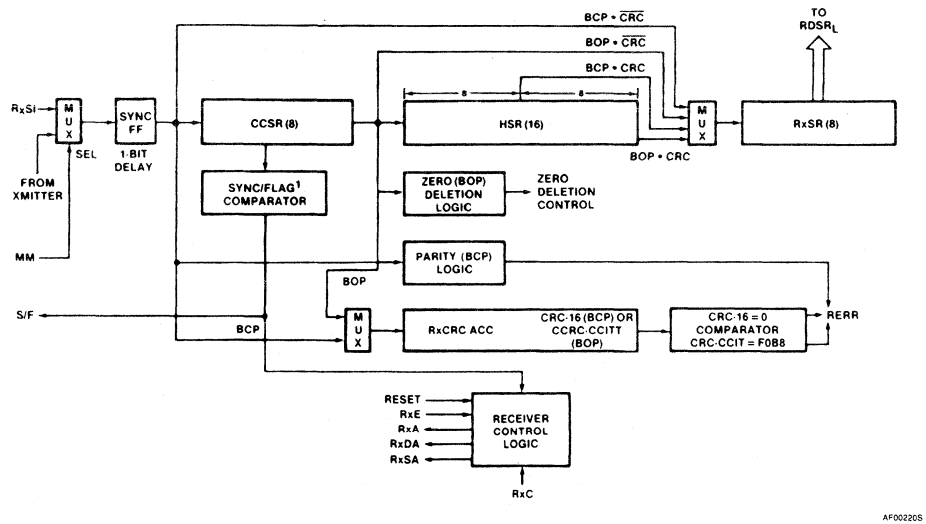
OPERATION	BIT PATTERN	FUNCTION
BOP		
FLAG	01111110	Frame message
ABORT	11111111 generation 01111111 detection	Terminate communication
GA	01111111	Terminate loop mode repeater function
Address	$(PCSAR_L)$[1]	Secondary station address
BCP		
SYNC	$(PCSAR_L)$ or (TxDB)[2] generation	Character synchronization

NOTES:
1. ( ) = contents of.
2. For IDLE = 0 or 1 respectively.



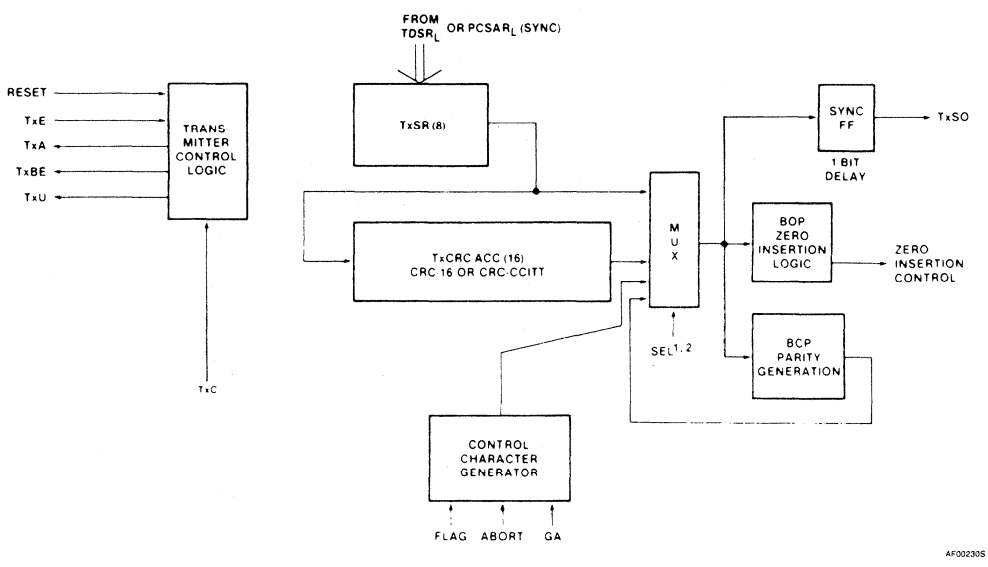
NOTE:
Refer to Register Formats for mnemonics and description.

Figure 1. Short Form Register Bit Formats



NOTES:
1. Detected in SYNC FF and 7 MS bits of CCSR.
2. In BOP mode, a minimum of two data characters must be received to turn the receiver active.

Figure 2. MPCC Receiver Data Path



NOTES:
1. TxCRC selected if TEOM = 1 and the last data character has been shifted out of TxSR.
2. In BCP parity selected will be generated after each character is shifted out of TxSR.

Figure 3. MPCC Transmitter Data Path

RECEIVER OPERATION

General

After initializing the parameter control registers (PCSAR and PCR), the RxE input must be set high to enable the receiver data path. The serial data on the RxSI is synchronized and shifted into an 8-bit Control Character Shift Register (CCSR) on the rising edge of RxC. A comparison between CCSR contents and the FLAG (BOP) or SYNC (BCP) character is made until a match is found. At that time, the S/F output is asserted for one RxC time and the 16-bit Holding Shift Register (HSR) is enabled. The receiver then operates as described below.

BOP Operation

A flowchart of receiver operation in BOP mode appears in figure 4. Zero deletion (after five ones are received) is implemented on the received serial data so that a data character will not be interpreted as a FLAG, ABORT, or GA. Bits following the FLAG are shifted through the CCSR, HSR, and into the Receiver Shift Register (RxSR). A character will be assembled in the RxSR and transferred to the $RDSR_L$ for presentation to the processor. At that time the RxDA output will be asserted and the processor must take the character no later than one RxC time after the next character is assembled in the RxSR. If not, an overrun ($RDSR_{11} = 1$) will occur and succeeding characters will be lost.

The first character following the FLAG is the secondary station address. If the MPCC is a secondary station ($PCSAR_{12} = 1$), the contents of RxSR are compared with the address stored in $PCSAR_L$. A match indicates the forthcoming message is intended for the station; the RxA output is asserted, the character is loaded into $RDSR_L$, RxDA is asserted and the Receive Start of Message bit (RSOM) is set. No match indicates that another station is being addressed and the receiver searches for the next FLAG.

If the MPCC is a primary station, ($PCSAR_{12} = 0$), no secondary address check is made; RxA is asserted and RSOM is set once the first non-FLAG character has been loaded into $RDSR_L$ and RxDA has been asserted. Extended address field can be supported by software if $PCSAR_{12} = 0$.

When the 8 bits following the address character have been loaded into $RDSR_L$ and RxDA has been asserted, RSOM will be cleared. The processor should read this 8-bit character and interpret it as the Control field.

Received serial data that follows is read and interpreted as the information field by the processor. It will be assembled into character lengths as specified by PCR_{8-10}. As before, RxDA is asserted each time a character has been transferred into $RDSR_L$ and is cleared

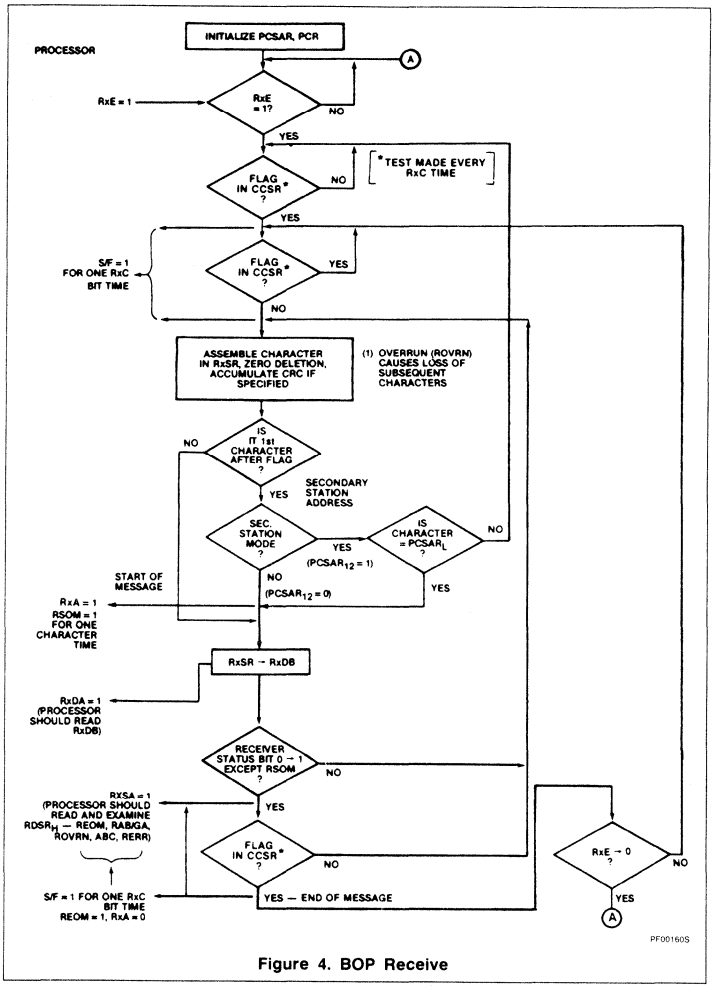


Figure 4. BOP Receive

when $RDSR_L$ is read by the processor. $RDSR_H$ should only be read when RxSA is asserted. This occurs on a zero to one transition of any bit in $RDSR_H$ except for RSOM. RxSA and all bits in $RDSR_H$ except RSOM are cleared when $RDSR_H$ is read. The processor should check $RDSR_{9-15}$ each time RxSA is asserted. If $RDSR_9$ is set, then $RDSR_{12-15}$ should be examined.

Receiver character length may be changed dynamically in response to RxDA: read the character in RxDB and write the new character length into RxCL. The character length will be changed on the next receiver character boundary. A received residual (short) character will be transferred into RxDB after the previous character in RxDB has been read, i.e. there will not be an overrun. In general the last two characters are protected from overrun.

The CRC-CCITT, if specified by $PCSAR_{8-10}$, is accumulated in RxCRC on each character following the FLAG. When the closing FLAG is detected in the CCSR, the received CRC is in the 16-bit HSR. At that time, the Receive End of Message bit (REOM) will be set; RxSA and RxDA will be asserted. The processor should read the last data character in $RDSR_L$ and the receiver status in $RDSR_{9-15}$. If $RDSR_{15} = 1$, there has been a transmission error; the accumulated CRC-CCITT is incorrect. If $RDSR_{12-14} \neq 0$, the last data charac-

ter is not of prescribed length. Neither the received CRC nor closing FLAG are presented to the processor. The processor may drop RxE or leave it active at the end of the received message.

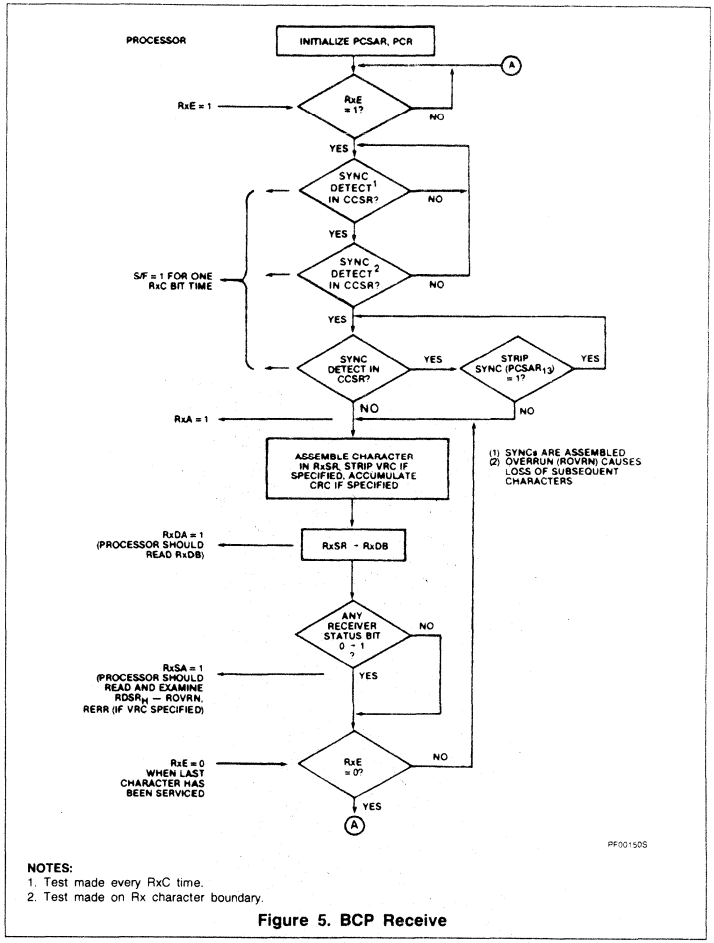
BCP Operation

The operation of the receiver in BCP mode is shown in figure 5. The receiver initially searches for two successive SYNC characters, of length specified by PCR_{8-10}, that match the contents of $PCSAR_L$. The next non-SYNC character or next SYNC character, if stripping is not specified ($PCSAR_{13} = 0$), causes RxA to be asserted and enables the receiver data path. Once enabled, all characters are assembled in RxSR and loaded into $RDSR_L$. RxDA is active when a character is available in $RDSR_L$. RxSA is active on a 0 to 1 transition of any bit in $RDSR_H$. The signals are cleared when $RDSR_L$ or $RDSR_H$ are read respectively.

If CRC-16 error control is specified by $PCSAR_{8-10}$, the processor must determine the last character received prior to the CRC field. When that character is loaded into $RDSR_L$ and RxDA is asserted, the received CRC will be in CCSR and HSR_L. To check for a transmission error, the processor must read the receiver status ($RDSR_H$) and examine $RDSR_{15}$. This bit will be set for one character time if an error free message has been received. If $RDSR_{15} = 0$, the CRC-16 is in error. The state of $RDSR_{15}$ in BCP CRC mode does not set RxSA. Note that this bit should be examined only at the end of a message. The accumulated CRC will include all characters starting with the first non-SYNC character if $PCSAR_{13} = 1$, or the character after the opening two SYNCs if $PCSAR_{13} = 0$. This necessitates external CRC generation/checking when supporting IBM's BISYNC. This can be accomplished using the Signetics SCN2653 Polynomial Generator/Checker. See Typical Applications.

If VRC has been selected for error control, parity (odd or even) is regenerated on each character and checked when the parity bit is received. A discrepancy causes $RDSR_{15}$ to be set and RxSA to be asserted. This must be sensed by the processor. The received parity bit is stripped before the character is presented to the processor.

When the processor has read the last character of the message, it should drop RxE which disables the receiver logic and initializes all receiver registers and timing.



NOTES:
1. Test made every RxC time.
2. Test made on Rx character boundary.

Figure 5. BCP Receive

TRANSMITTER OPERATION

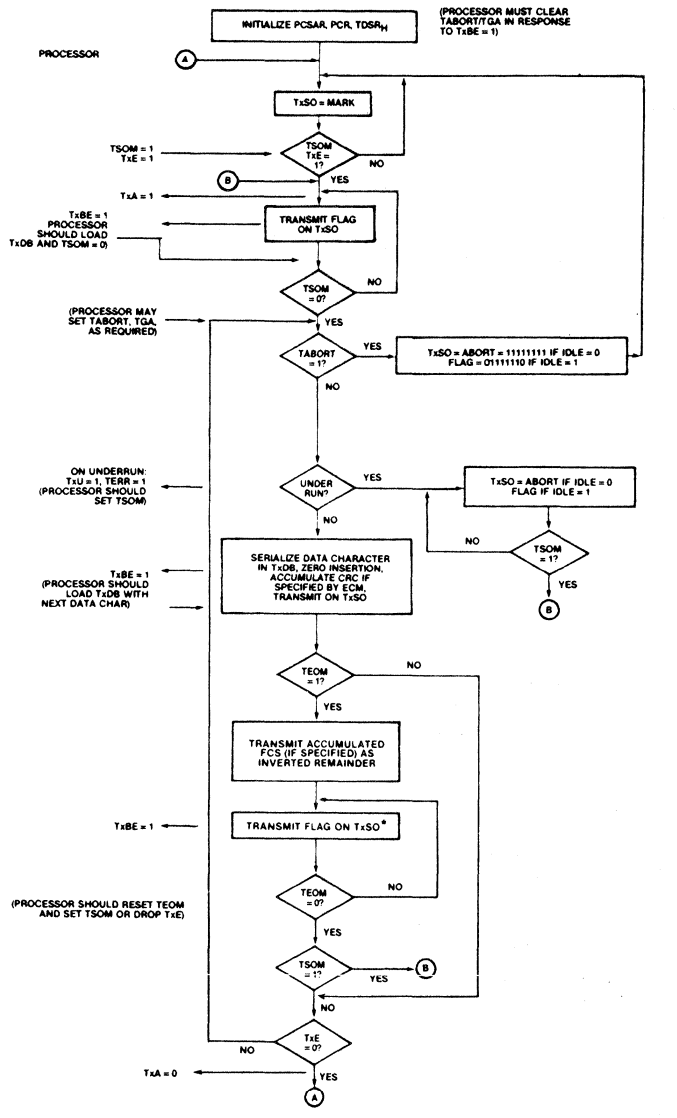
General

After the parameter control registers (PCSAR and PCR) have been initialized, TxSO is held at mark until TSOM ($TDSR_8$) is set and TxE is raised. Then, transmitter operation depends on protocol mode.

BOP Operation

Transmitter operation for BOP is shown in figure 6. A FLAG is sent after the processor sets the Transmit Start of Message bit (TSOM) and raises TxE. The FLAG is used to synchronize the message that follows. TxA will also be asserted. When TxBE is asserted by the MPCC, the processor should load $TDSR_L$ with the first character of the message. TSOM should be cleared at the same time $TDSR_L$ is loaded (16-bit data bus) or immediately thereafter (8-bit data bus). FLAGs are sent as long as TSOM = 1. For counting the number of FLAGs, the processor should reassert TSOM in response to the assertion of TxBE.

All succeeding characters are loaded into $TDSR_L$ by the processor when TxBE = 1. Each character is serialized in TxSR and transmitted on TxSO. Internal zero insertion logic stuffs a "0" into the serial bit stream after five successive "1s" are sent. This insures a data character will not match a FLAG, ABORT, or GA reserved control character. As each character is transmitted, the Frame Check Sequence (FCS) is generated as specified by Error Control Mode ($PCSAR_{8-10}$). The FCS should be the CRC-CCITT polynomial ($X^{16} + X^{12} + X^5 + 1$) preset to 1s. If an underrun occurs (processor is not keeping up with the transmitter), TxU and TERR ($TDSR_{15}$) will be asserted with ABORT or FLAG used as the TxSO line fill depending on the state of IDLE ($PCSAR_{11}$). The proces-



*GA will be transmitted if TGA is set together with TEOM.

Figure 6. BOP Transmit

sor must set TSOM to reset the underrun condition. To retransmit the message, the processor should proceed with the normal start of message sequence.

A residual character of 1 to 7 bits may be transmitted at the end of the information field. In response to TxBE, write the residual character length into TxCL and load TxDB with the residual character. Dynamic alteration of character length should be done in exactly the same sequence. The character length will be changed on the next transmit character boundary.

After the last data character has been loaded into $TDSR_L$ and sent to TxSR (TxBE = 1), the processor should set TEOM ($TDSR_9$). The MPCC will finish transmitting the last character followed by the FCS and the closing FLAG. The processor should clear TEOM and drop TxE when the next TxBE is asserted. This corresponds to the start of closing FLAG transmission. When TxE has been dropped. TxA will be low 1½ bit times after the last bit of the closing FLAG has been transmitted. TxSO will be marked after the closing FLAG has been transmitted.

If TxE and TEOM are high, the transmitter continues to send FLAGs. The processor may initiate the next message by resetting TEOM and setting TSOM, or by loading $TDSR_L$ with a data character and then simply resetting TSOM (without setting TSOM).

BCP Operation

Transmitter operation for BCP mode is shown in figure 7. TxA will be asserted after TSOM = 1 and TxE is raised. At that time SYNC characters are sent from $PCSAR_L$ or $TDSR_L$ (IDLE = 0 or 1) as long as TSOM = 1. TxBE is asserted at the start of transmission of the first SYNC character. For counting the number of SYNCs, the processor should reassert TSOM in response to the assertion of TxBE. When TSOM = 0 transmission is from $TDSR_L$, which must be loaded with characters from the processor each time TxBE is asserted. If this loading is delayed for more than one character time, an underrun results: TxU and TERR are asserted and the TxSO line fill depend on IDLE ($PCSAR_{11}$). The processor must set TSOM and retransmit the message to recover. This is not compatible with IBM's BISYNC, so that the user must not underrun when supporting that protocol.

CRC-16, if specified by $PCSAR_{8-10}$, is generated on each character transmitted from $TDSR_L$ when TSOM = 0. The processor must set TEOM = 1 after the last data character has been sent to TxSR (TxBE = 1). The MPCC will finish transmitting the last data character and the CRC-16 field before sending SYNC characters which are transmitted as long as TEOM = 1. If SYNCs are not desired after CRC-16 transmission, the processor should clear TEOM and lower TxE when the TxBE corresponding to the start of CRC-16 transmission is asserted. When TEOM = 0, the line is marked and a new

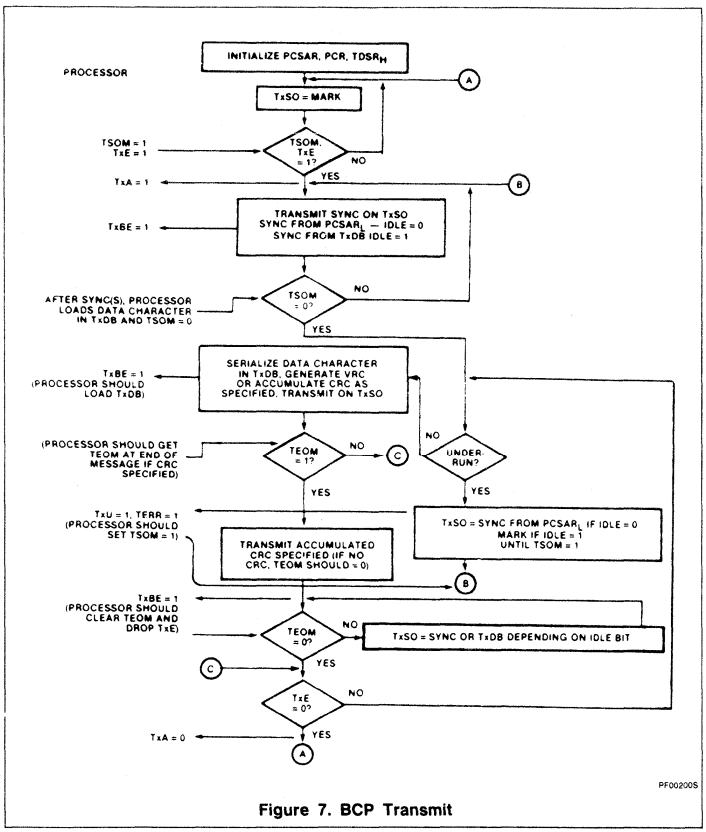


Figure 7. BCP Transmit

message may be initiated by setting TSOM and raising TxE.

If VRC is specified, it is generated on each data character and the data character length must not exceed 7 bits. For software LRC or CRC, TEOM should be set only if SYNC's are required at the end of the message block.

Special Case

The capability to transmit 16 spaces is provided for line turnaround in half duplex mode or for a control recovery situation. This is achieved by setting TSOM and TEOM, clearing TEOM when TxBE = 1, and proceeding as required.

PROGRAMMING

Prior to initiating data transmission or reception, PCSAR and PCR must be loaded with control information from the processor. The contents of these registers (see Register Format section) will configure the MPCC for the user's specific data communication environment. These registers should be loaded during power-on initialization and after a reset operation. They can be changed at any time that the respective transmitter or receiver is disabled.

The default value for all registers is zero. This corresponds to BOP, primary station mode, 8-bit character length, FCS = CRC-CCITT preset to 1s.

For BOP mode the character length register (PCR) may be set to the desired values during system initialization. The address and control fields will automatically be 8-bits. If a residual character is to be transmitted, TxCL should be changed to the residual character length prior to transmission of that character.

DATA BUS CONTROL

The processor must set up the MPCC register address (A2 – A0), chip enable (CE), byte select (BYTE), and read/write ($\overline{R}$/W) inputs before each data bus transfer operation.

During a read operation ($\overline{R}$/W = 0), the leading edge of DBEN will initiate an MPCC read cycle. The addressed register will place its contents on the data bus. If BYTE = 1, the 8-bit byte is placed on DB15-08 or DB07-00 depending on the H/L status of the register addressed. Unused bits in RDSR$_L$ are zero. If BYTE = 0, all 16 bits (DB15-00) contain MPCC information. The trailing edge of DBEN will reset RxDA and/or RxSA if RDSR$_L$ or RDSR$_H$ is addressed respectively.

DBEN acts as the enable and strobe so that the MPCC will not begin its internal read cycle until DBEN is asserted.

During a write operation ($\overline{R}$/W = 1), data must be stable on DB$_{15-08}$ and/or DB$_{07-00}$ prior to the leading edge of DBEN. The stable data is strobed into the addressed register by DBEN. TxBE will be cleared if the addressed register was TDSR$_H$ or TDSR$_L$.

Product Specification

Table 4. MPCC REGISTER ADDRESSING

	A2	A1	A0	REGISTER
BYTE = 0	**16-BIT DATA BUS = DB_{15} - DB_{00}**			
	0	0	X	RDSR
	0	1	X	TDSR
	1	0	X	PCSAR
	1	1	X	PCR*
BYTE = 1	**8-BIT DATA BUS = DB_{7-0} or DB_{15-8}****			
	0	0	0	$RDSR_L$
	0	0	1	$RDSR_H$
	0	1	0	$TDSR_L$
	0	1	1	$TDSR_H$
	1	0	0	$PCSAR_L$
	1	0	1	$PCSAR_H$
	1	1	0	PCR_L*
	1	1	1	PCR_H

NOTES:
* PCR lower byte does not exist. It will be all "0"s when read.
** Corresponding high and low order pins must be tied together.

Table 5. PARAMETER CONTROL REGISTER (PCR) - (R/W)

BIT	NAME	MODE	FUNCTION
00 - 07	Not Defined		
08 - 10	RxCL	BOP/BCP	Receiver character length is loaded by the processor when RxCLE = 0. The character length is valid after transmission of single byte address and control fields have been received.

10	9	8	Char length (bits)
0	0	0	8
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

11	RxCLE	BOP/BCP	Receiver character length enable should be zero when the processor loads RxCL. The remaining bits of PCR are not affected during loading. Always 0 when read.
12	TxCLE	BOP/BCP	Transmitter character length enable should be zero when the processor loads TxCL. The remaining bits of PCR are not affected during loading. Always 0 when read.
13 - 15	TxCL	BOP/BCP	Transmitter character length is loaded by the processor when TxCLE = 0. Character bit length specification format is identical to RxCL. It is valid after transmission of single byte address and control fields.

Product Specification

Table 6. PARAMETER CONTROL SYNC/ADDRESS REGISTER (PCSAR)-(R/W)

BIT	NAME	MODE	FUNCTION
00 – 07	S/AR	BOP	SYNC/address register. Contains the secondary station address if the MPCC is a secondary station. The contents of this register is compared with the first received non-FLAG character to dertermine if the message is meant for this station.
		BCP	SYNC character is loaded into this register by the processor. It is used for receive and transmit bit synchronization with bit length specified by RxCL and TxCL.
08 – 10	ECM	BOP/BCP	**Error Control Mode** / **10** / **9** / **8** / **Suggested Mode** / **Char. length** CRC-CCITT preset to 1's / 0 / 0 / 0 / BOP / 1 – 8 CRC-CCITT preset to 0's / 0 / 0 / 1 / BCP / 8 Not used / 0 / 1 / 0 / — / CRC-16 preset to 0's / 0 / 1 / 1 / BCP / 8 VRC odd / 1 / 0 / 0 / BCP / 5 – 7 VRC even / 1 / 0 / 1 / BCP / 5 – 7 Not used / 1 / 1 / 0 / — / No error control / 1 / 1 / 1 / BCP/BOP / 5 – 8 ECM should be loaded by the processor during initialization or when both data paths are idle.
11	IDLE		Determines line fill character to be used if transmitter underrun occurs (TxU asserted and TERR set) and transmission of special characters for BOP/BCP.
		BOP	IDLE = 0, transmit ABORT characters during underrun and when TABORT = 1. IDLE = 1, transmit FLAG characters during underrun and when TABORT = 1.
		BCP	IDLE = 0 transmit initial SYNC characters and underrun line fill characters from the S/AR. IDLE = 1 transmit initial SYNC characters from TxDB and marks TxSO during underrun.
12	SAM	BOP	Secondary Address Mode = 1 if the MPCC is a secondary station. This facilitates automatic recognition of the received secondary station address. When transmitting, the processor must load the secondary address into TxDB. SAM = 0 inhibits the received secondary address comparison which serves to activate the receiver after the first non-FLAG character has been received.
13	SS/GA		Strip SYNC/Go Ahead. Operation depends on mode.
		BOP	SS/GA = 1 is used for loop mode only and enables GA detection. When a GA is detected as a closing character, REOM and RAB/GA will be set and the processor should terminate the repeater function. SS/GA = 0 is the normal mode which enables ABORT detection. It causes the receiver to terminate the frame upon detection of an ABORT or FLAG.
		BCP	SS/GA = 1, causes the receiver to strip SYNC's immediately following the first two SYNC's detected. SYNC's in the middle of a message will not be stripped. SS/GA = 0, presents any SYNC's after the initial two SYNC's to the processor.
14	PROTO		Determines MPCC Protocol mode
		BOP BCP	PROTO = 0 PROTO = 1
15	APA	BOP	All parties address. If this bit is set, the receiver data path is enabled by an address field of '11111111' as well as the normal secondary station address.

Product Specification

Table 7. TRANSMIT DATA/STATUS REGISTER (TDSR) (R/W EXCEPT TDSR15)

BIT	NAME	MODE	FUNCTION
00 – 07	TxDB	BOP/BCP	Transmit data buffer. Contains processor loaded characters to be serialized in TxSR and transmitted on TxSO.
08	TSOM		Transmitter start of message. Set by the processor to initate message transmission provided TxE = 1.
		BOP	TSOM = 1 generates FLAGs. When TSOM = 0 transmission is from TxDB and FCS generation (if specified) begins. FCS, as specified by $PCSAR_{8-10}$, should be CRC-CCITT preset to 1's.
		BCP	TSOM = 1 generates SYNCs from $PCSAR_L$ or transmits from TxDB for IDLE = 0 or 1 respectively. When TSOM = 0 transmission is from TxDB and CRC generation (if specified) begins.
09	TEOM		Transmit end of message. Used to terminate a transmitted message.
		BOP	TEOM = 1 causes the FCS and the closing FLAG to be transmitted following the transmission of the data character in TxSR. FLAGs are transmitted until TEOM = 0. ABORT or GA are transmitted if TABORT or TGA are set when TEOM = 1.
		BCP	TEOM = 1 causes CRC-16 to be transmitted (if selected) followed by SYNCs from $PCSAR_L$ or TxDB (IDLE = 0 or 1). Clearing TEOM prior to the end of CRC-16 transmission (when TxBE = 1) causes TxSO to be marked following the CRC – 16. TxE must be dropped before a new message can be initiated. If CRC is not selected, TEOM should not be set.
10	TABORT	BOP	Transmitter abort = 1 will cause ABORT or FLAG to be sent (IDLE = 0 or 1) after the current character is transmitted. (ABORT = 11111111)
11	TGA	BOP	Transmit go ahead (GA) instead of FLAG when TEOM = 1. This facilitates repeater termination in loop mode. (GA = 01111111)
12 – 14	Not Defined		
15	TERR	Read only	Transmitter error = 1 indicates the TxDB has not been loaded in time (one character time-½ TxC period after TxBE is asserted) to maintain continuous transmission. TxU will be asserted to inform the processor of this condition. TERR is cleared by setting TSOM. See timing diagram.
		BOP	ABORT's or FLAG's are sent as fill characters (IDLE = 0 or 1)
		BCP	SYNC's or MARK's are sent as fill characters (IDLE = 0 or 1). For IDLE = 1 the last character before underrun is not valid.

Product Specification

Table 8. RECEIVER DATA/STATUS REGISTER (RDSR) - (READ ONLY)

BIT	NAME	MODE	FUNCTION
00 - 07	RxDB	BOP/BCP	Receiver data buffer. Contains assembled characters from the RxSR. If VRC is specified, the parity bit is stripped.
08	RSOM	BOP	Receiver start of message = 1 when a FLAG followed by a non-FLAG has been received and the latter character matches the secondary station if SAM = 1. RxA will be asserted when RSOM = 1. RSOM resets itself after one character time and has no affect on RxSA.
09	REOM	BOP	Receiver end of message = 1 when the closing FLAG is detected and the last data character is loaded into RxDB or when an ABORT/GA character is received. REOM is cleared on reading $RDSR_H$, reset operation, or dropping of RxE.
10	RAB/GA	BOP	Received ABORT or GA character = 1 when the receiver senses an ABORT character if SS/GA = 0 or a GA character if SS/GA = 1. RAB/GA is cleared on reading $RDSR_H$, reset operation, or dropping of RxE. A received abort does not set RxDA.
11	ROR	BOP/BCP	Receiver overrun = 1 indicates the processor has not read last character in the RxDB within one character time + ½ RxC period after RxDA is asserted. Subsequent characters will be lost. ROR is cleared on reading $RDSR_H$, reset operation, or dropping of RxE.
12 - 14	ABC	BOP	Assembled bit count. Specifies the number of bits in the last received data character of a message and should be examined by the processor when REOM = 1(RxDA and RxSA asserted). ABC = 0 indicates the message was terminated (by a flag or GA) on a character boundary as specified by $PCR_{8\text{-}10}$. Otherwise, ABC = number of bits in the last data character. ABC is cleared when $RDSR_H$ is read, reset operation, or dropping RxE. The residual character is right justified in $RDSR_L$.
15	RERR	BOP/BCP	Receiver error indicator should be examined by the processor when REOM = 1 in BOP, or when the processor determines the last data character of the message in BCP with CRC or when RxSA is set in BCP with VRC. CRC-CCITT preset to 1's/0's as specified by $PCSAR_{8-10}$: RERR = 1 indicates FCS error (CRC ≠ F0B8 or ≠ 0) RERR = 0 indicates FCS received correctly (CRC = F0B8 or = 0) CRC-16 preset to 0's on 8-bit characters specified by $PSCAR_{8-10}$: RERR = 1 indicates CRC-16 received correctly (CRC = 0). RERR = 0 indicates CRC-16 error (CRC≠0) VRC specified by $PCSAR_{8-10}$: RERR = 1 indicates VRC error RERR = 0 indicates VRC is correct.

ABSOLUTE MAXIMUM RATINGS[1]

	PARAMETER	RATING	UNIT
T_A	Operating ambient temperature[2]	Note 4	°C
T_{STG}	Storage temperature	-65 to +150	°C
	Input or output voltages with respect to GND[3]	-0.3 to +15	V
V_{CC}	With respect to GND	-0.3 to +7	V

Product Specification

DC ELECTRICAL CHARACTERISTICS[4,5]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	I_{OL} = 1.6mA			0.4	
V_{OH}	High	I_{OH} = −100μA	2.4			
I_{CC}	Power supply current	V_{CC} = 5.25V, T_A = 0°C			150	mA
Leakage current						μA
I_{IL}	Input	V_{IN} = 0 to 5.25V			10	
I_{OL}	Output	V_{OUT} = 0 to 5.25V			10	
Capacitance						pF
C_{IN}	Input	V_{IN} = 0V, f = 1MHz			20	
C_{OUT}	Output	V_{OUT} = 0V, f = 1MHz			20	

AC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		1MHz CLOCK VERSION			2MHz CLOCK VERSION			UNIT
		Min	Typ	Max	Min	Typ	Max	
Set-up and hold time								ns
t_{ACS}	Address/control set-up	50			50			
t_{ACH}	Address/control hold	0			0			
t_{DS}	Data bus set-up (write)	50			50			
t_{DH}	Data bus hold (write)	0			0			
t_{RXS}	Receiver serial data set-up	150			150			
t_{RxH}	Receiver serial data hold	150			150			
Pulse width								ns
t_{RES}	RESET	250			250			
t_{DBEN}	DBEN	250		m[7]	200		m[7]	
Delay Time								ns
t_{DD}	Data bus (read)			200			170	
t_{TxD}	Transmit serial data			325			250	
t_{DBEND}	DBEN to DBEN delay	200			200			
t_{DF}	Data bus float time (read)			150			150	ns
f	Clock (RxC,TxC) frequency			1.0			2.0	MHz
t_{CLK1}	Clock high (MM = 0)	340			165			ns
t_{CLK2}	Clock high (MM = 1)	490			240			
t_{CLK0}	Clock low	490			240			

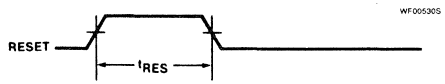
NOTES:

1. Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation sections of this specification is not implied.
2. For operating at elevated temperatures the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. See ordering code table for applicable temperature range and operating supply range.
5. All voltage measurements are referenced to ground. All time measurements are at 0.8V or 2.0V. Input voltage levels for testing are 0.4V and 2.4V.
6. Output load C_L = 100pF.
7. m = TxC low and applies to writing to $TDSR_H$ only.

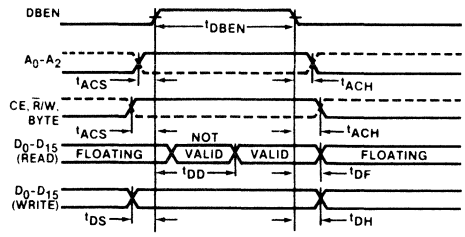
Product Specification

TIMING DIAGRAMS

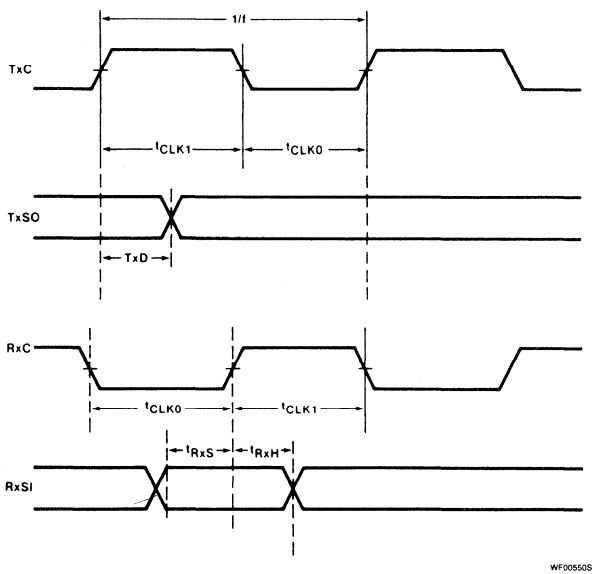
RESET



READ AND WRITE DATA BUS



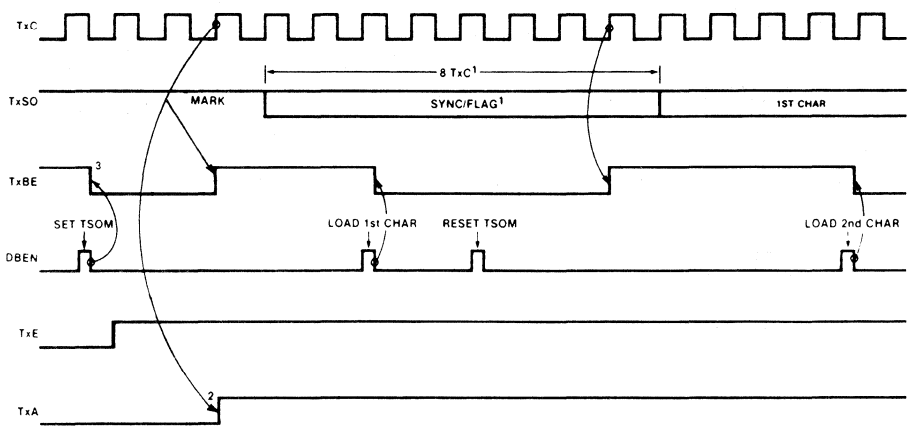
CLOCK



Product Specification

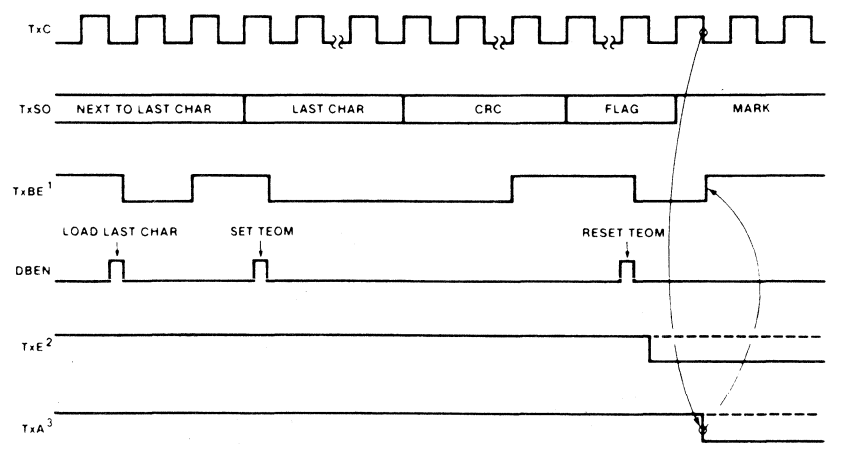
TIMING DIAGRAMS (Continued)

TRANSMIT – START OF MESSAGE



NOTES:
1. SYNC may be 5 to 8 bits and will contain parity bit as specified.
2. TxA goes high relative to TxC rising edge after TSOM has been set and TxE has been raised.
3. TxBE goes low relative to DBEN falling edge on the first write transfer into TDSR. It is reasserted 1 TxC time before the first bit of the transmitted SYNC/FLAG. TxBE then goes low relative to DBEN falling edge when writing into $TDSR_H$ and/or $TDSR_L$. It is reasserted on the rising edge of the TxC that corresponds to the transmission of the last bit of each character, except in BOP mode when the CRC is to be sent as the next character (see Transmit Timing – End of Message).

TRANSMIT — END OF BOP MESSAGE

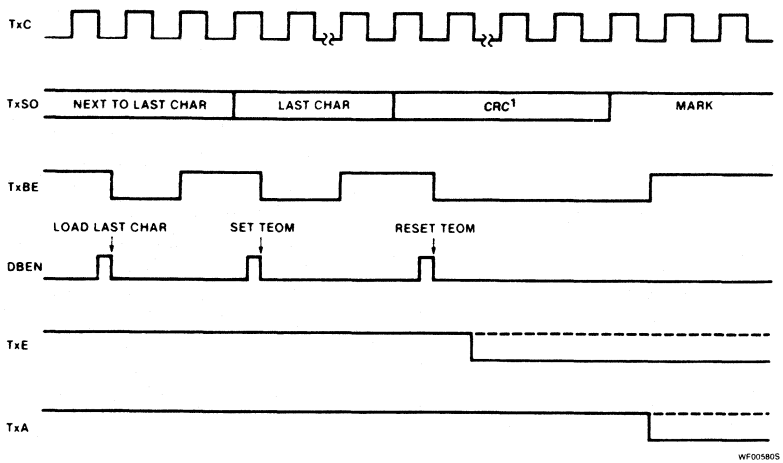


NOTES:
1. TxBE goes low relative to the falling edge of DBEN corresponding to loading $TDSR_{H/L}$. It goes high one TxC before character transmission begins and also when TxA has been dropped.
2. TxE can be dropped before resetting TEOM if TxBE (corresponding to the closing FLAG) is high. Alternatively TxE can remain high and a new message initiated.
3. TxA goes low after TxE has been dropped and 1½ TxC's after the last bit of the closing FLAG has been transmitted.

Product Specification

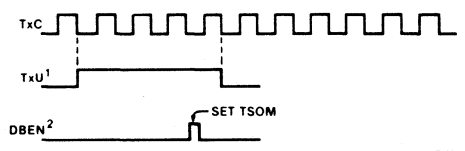
TIMING DIAGRAMS (Continued)

TRANSMIT TIMING — END OF BCP MESSAGE



NOTES:
1. When SCN2652 generated CRC is not required, TEOM should only be set if SYNCs are to follow the message block. In that case, TxE should be dropped in response to TxBE (which corresponds to the start of transmission of the last character). When CRC is required, TxE must be dropped before CRC transmission is complete. Otherwise, the contents of TxDB will be shifted out on TxSO. This facilitates transmission of contiguous messages.

TRANSMIT UNDERRUN

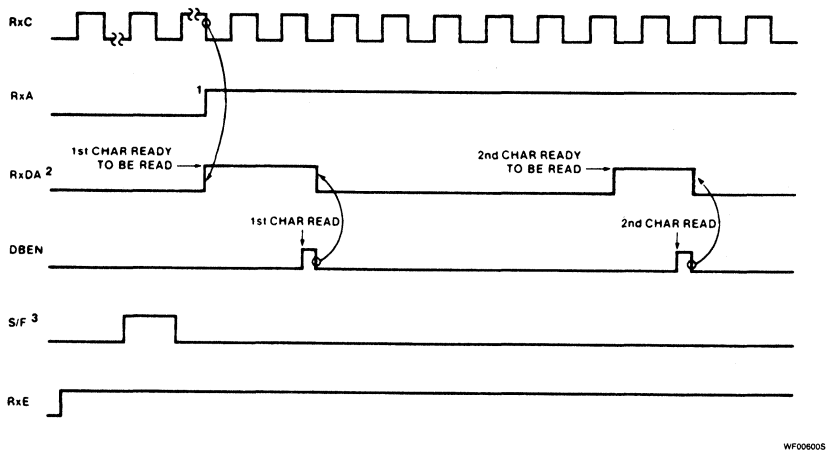


NOTES:
1. TxU goes active relative to TxC falling edge if TxBE has not been serviced after n-½ TxC times (where n = transmit character length). TxU is reset on the TxC falling edge following assertion of the TSOM command.
2. An underrun will occur at the next character boundary if TEOM is reset and the transmitter remains enabled, unless the TSOM command is asserted or a character is loaded into the TxDB.

Product Specification

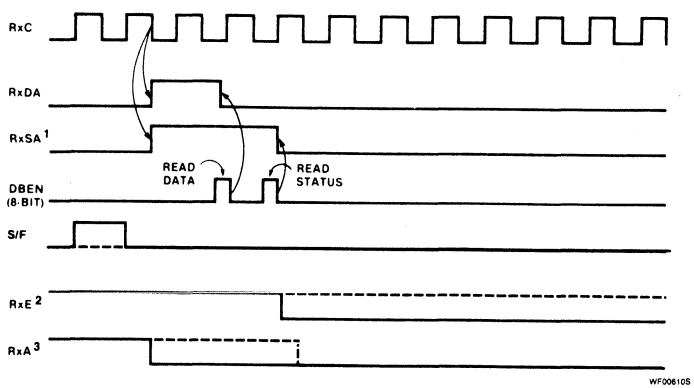
TIMING DIAGRAMS (Continued)

RECEIVE — START OF MESSAGE



NOTES:
1. RxA goes high relative to falling edge of RxC when RxE is high and: a. A data character following two SYNC's is in RxDB (BCP mode). b. Character following FLAG is in RxDB (BOP primary station mode). c. Character following FLAG is in RxDB and character matches the secondary station address or all parties address (BOP secondary station mode).
2. RxDA goes high on RxC falling edge when a character in RxDB is ready to be read. It comes up before RxSA and goes low on the falling edge of DBEN when RxDB is read.
3. S/F goes high relative to rising edge of RxC anytime a SYNC (BCP) or FLAG (BOP) is detected.

RECEIVE END OF MESSAGE

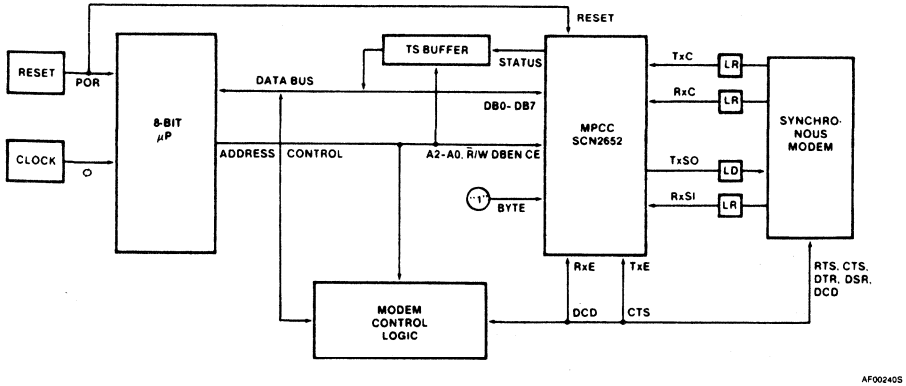


NOTES:
1. At the end of a BOP message, RxSA goes high when FLAG detection (S/F 1) forces REOM to be set. Processor should read the last data character ($RDSR_L$) and status ($RDSR_H$) which resets RxDA and RxSA respectively. For BCP end of message, RxSA may not be set and S/F = 0. The processor should read the last data character and status.
2. RxE must be dropped for BCP with non-contiguous messages. It may be left on at the end of a BOP message (see BOP Receive Operation).
3. RxA is reset relative to the falling edge of RxC after the closing FLAG of a BOP message (REOM = 1 and RxSA active.) or when RxE is dropped.

Product Specification

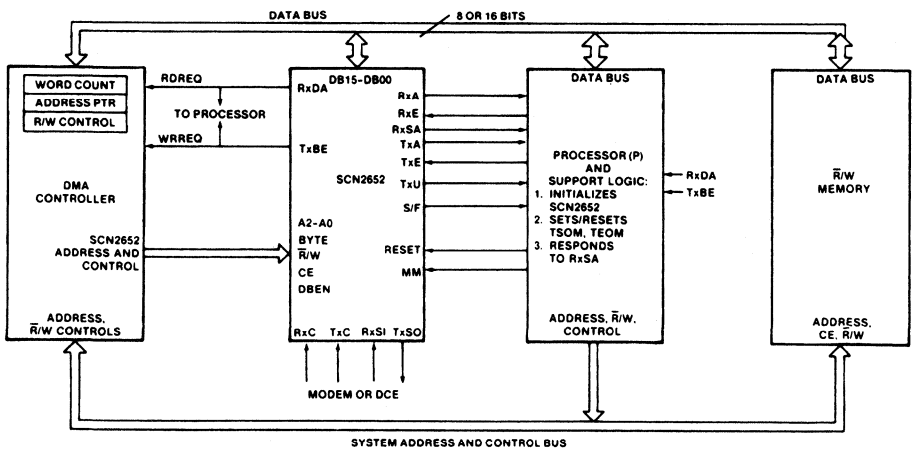
TYPICAL APPLICATIONS

SCN2652 MPCC MICROPROCESSOR INTERFACE



NOTES:
1. Possible μP interrupt requests are: RxDA RxSA TxBE TxU
2. Other SCN2652 status signals and possible uses are: S F line idle indicator, frame delimiter. RxA handshake on RxE, line turn around control. TxA handshake on TxE, line turn around control.
3. Line drivers/receivers (LD/LR) convert EIA to TTL voltages and vice-versa.
4. RTS should be dropped after the CRC (BCP) or FLAG (BOP) has been transmitted. This forces CTS low and TxE low.
5. Corresponding high and low order bits of DB must be OR tied.

DMA/PROCESSOR INTERFACE

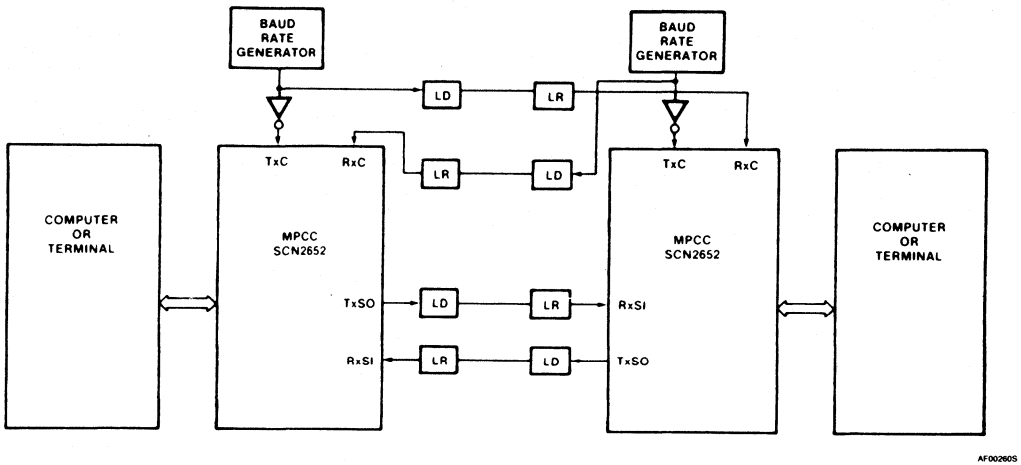


For non-DMA operation TxBE and RxDA are sent to the processor which then loads or reads data characters as required.

Product Specification

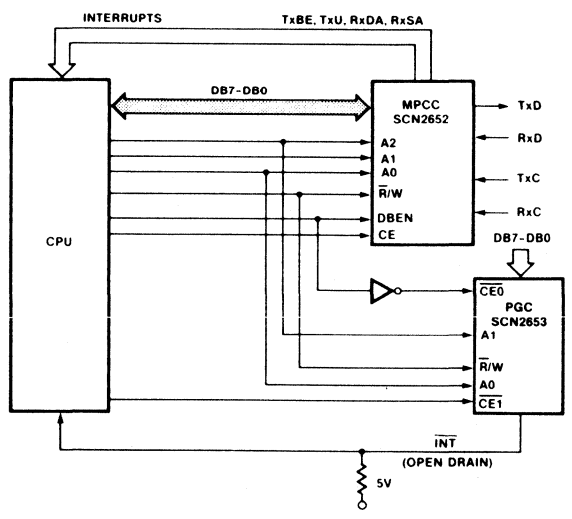
TYPICAL APPLICATIONS (Continued)

CHANNEL INTERFACE



No Modem — DC Baseband Transmission

SCN2652/SCN2653 INTERFACE TYPICAL PROTOCOLS:
BISYNC, DDCMP, SDLC, HDLC



Polynomial Generator Checker (PGC)

Originally published by Signetics February 1985

Product Specification

DESCRIPTION

The Signetics SCN2653/68653 Polynomial Generator Checker (PGC) is a polynomial generator checker/character comparator circuit that complements a receiver/transmitter (R/T or USART/USRT/UART) in the support of character oriented data link controls. Table 1 defines many of the more commonly used PGC terms and abbreviations.

Parallel data characters transferred between the CPU and R/T are monitored by the PGC which performs block check character (BCC) and parity (VRC) generation/checking, single character detection, and two character sequence detection. Since the PGC operates on parallel characters, the data transmission format may be serial (synchronous or asynchronous) or parallel.

There are four modes of BCC accumulation and each mode can select one of three polynomials to compute the BCC. In the BISYNC normal and transparent modes, the PGC determines which characters are to be accumulated and which characters are to be excluded from the accumulation. The block terminating characters and the initiation and termination of BISYNC transparent text can be detected and an interrupt generated. The single interrupt output represents the inclusive OR of four maskable status conditions.

In the automatic accumulation mode, all characters are accumulated while the single accumulate mode requires a specific accumulation command for each character to be accumulated.

Character accumulation control and character comparisons are facilitated by a character class array which places each of 128 characters into one of four character classes. The four classes are normal, SYN/BISYNC not included, block terminating character (BTC)/search character (SC), and secondary search character (SSC).

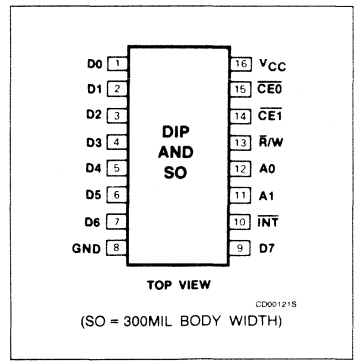
FEATURES

- **Parallel Block Check Character accumulation/checking: CRC-16, CRC-12, LRC-8**
- **BISYNC normal and transparent modes**
- **Automatic or single character accumulation modes**
- **Character detection - up to 128 characters**
- **Two character sequence detection; examples: DLE-STX, ACK0, ACK1, WACK, RVI, DISC, WBT**
- **6, 7, or 8-bit characters**
- **VRC generation/checking on data bus**
- **Four maskable interrupt conditions**
- **Four classes of characters**
- **Internal power-on reset**
- **Maximum character accumulation rate of 500kHz (4Mbps)**
- **Directly compatible with Signetics SCN2651, SCN2652 and SCN2661**
- **No system clock required**
- **TTL compatible inputs and outputs**
- **Single 5V supply**
- **16-pin dual in line package**

APPLICATIONS

- **Character oriented data link control:**
  - **dedicated to one USART/USRT**
  - **multiplexed among several USART/USRTs**
- **Automated BISYNC with 2661 (minimal software intervention)**
- **BCC and VRC generation/detection on a block of memory or peripheral data**
- **Programmable character array comparator**

PIN CONFIGURATION



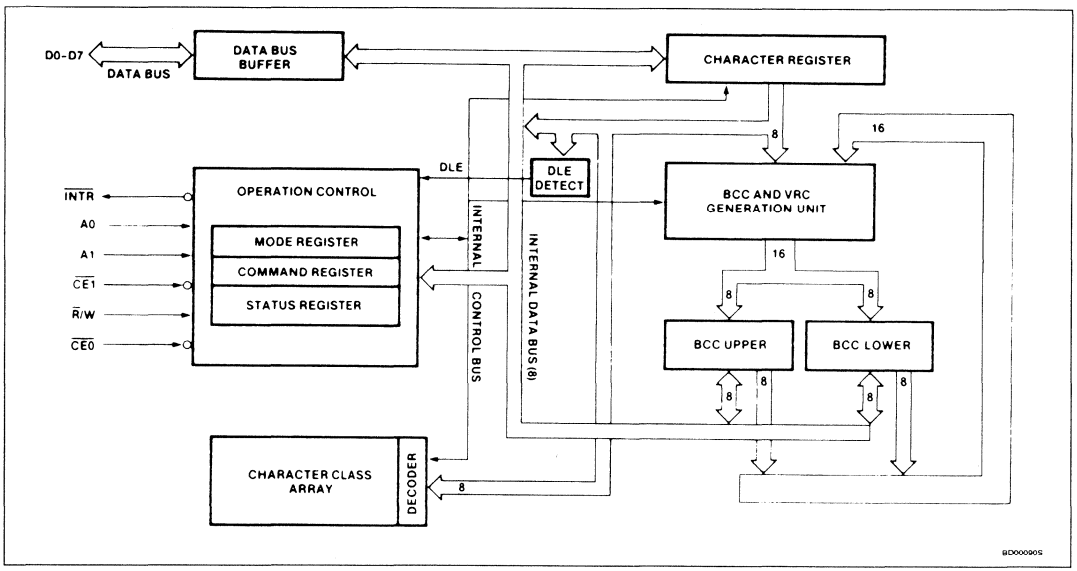
Product Specification

ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%, T_A = 0°C to +70°C
Ceramic DIP	SCN2653AC4I16
Plastic DIP	SCN2653AC4N16
Small Outline (SO)	SCN2653ACD16

NOTE:
SCN68653 is identical to SCN2653. Order using numbers shown above.

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{CC}	16	I	**+5V:** Power supply
GND	8	I	Ground
A1-A0	11,12	I	**Address Lines:** Used to select internal PGC registers or character class array.
R/W	13	I	**Read/Write:** Read command when low, write command when high.
CE0	15	I	**Chip Enable:** Connected to chip enable input of a receiver/transmitter (R/T) circuit. It is used to strobe data being transferred between the CPU and the R/T into the PGC character register.
CE1	14	I	**Chip Enable:** Used in conjunction with the R/W signal to enable the transfer of data between the PGC and the CPU or DMA controller and to initialize the PGC registers.
D7-D0	9,7-1	I/O	**Data Bus:** 8-bit three-state bidirectional bus used to transfer data to or from the PGC via CE0 or CE1. All data, mode words, command words, and status information are transferred on this bus. D0 is the least significant bit; D7 is the most significant bit.
INT	10	O	**Interrupt:** Open drain active low interrupt output that signals the CPU that one or more maskable conditions are true: BCC error, VRC error, BTC/SC detect, SSC detect. The true conditions can be determined by reading the status register which in turn deactivates INT. A power on, clear BCC, or master reset command causes INT to be inactive (high).

Product Specification

Additional PGC applications include off-line R/T operation where the BCC is generated on data not sent to the R/T, BCC multiplexing by sharing the PGC among several R/Ts and reading/writing the partial BCC accumulation on a character by character basis, VRC generation/checking on characters appearing on a bidirectional data bus, and programmable character comparisons or searches.

PGC operation is half duplex (either receive or transmit, one way or two way alternate). Full duplex (two way simultaneous) is achieved by using two PGCs. The device is directly compatible with the Signetics SCN2651 Programmable Communications interface (PCI) and SCN2661 Enhanced Programmable Communications interface (EPCI). When used in BISYNC modes with the SCN2661, software requirements are minimized by the SCN2653-SCN2661 control character comparisons, character sequence comparisons, and automatic DLE insertion/detection.

Other bus oriented R/Ts can be interfaced to the PGC with a minimum of external circuitry. See figure 1 for a typical system configuration.

This NMOS LSI circuit is TTL compatible, operates from a single + 5V supply and is contained in a 16 pin dual in line package.

BLOCK DIAGRAM

The PGC consists of six major sections. These are the operation control, character class array, DLE ROM, character register, BCC and parity generators, and BCC registers. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the CPU data bus via a data bus buffer.

Operation Control Unit

This functional block stores configuration and operation instructions from the CPU and generates appropriate signals to control the device operation. It also contains read and write circuits to permit communications between the CPU and the PGC registers via the data bus. The mode, command, and status registers are in this logic block.

Character Register

Characters to be considered for BCC generation, parity generation and checking, or character comparisons are loaded into this register by either $\overline{CE0}$ or $\overline{CE1}$. This register serves as an input to the BCC and VRC generator, where the accumulation and parity generation takes place. The character register also serves as the input for character class array and DLE comparisons.

Table 1. GLOSSARY

TERM/ABBREVIATION	DEFINITION
BCC	Block check character
BTC	Block terminating character
SC	Search character
SSC	Second search character (preceded by DLE)
CRC-16	$X^{16} + X^{15} + X^2 + 1$ divisor, dividend pre-cleared
CRC-12	$X^{12} + X^{11} + X^3 + X^2 + X + 1$ divisor, dividend pre-cleared
LRC-8	Horizontal parity on least significant 7 bits; vertical parity on most significant bit
VRC	Vertical redundancy check (character parity)
R/T	Receiver/transmitter circuit. Also known as USART/USRT/UART/PCI/MPCC
BISYNC	IBM binary synchronous communications (BSC), ANSI X3.28, ISO 1745
MSB	Most significant bit
LSB	Least significant bit
Rx	Receive
Tx	Transmit

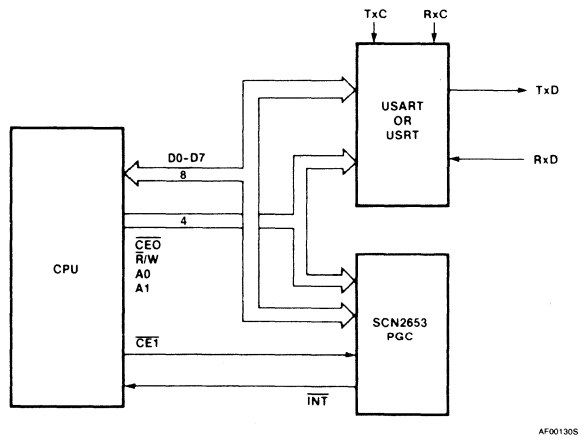


Figure 1. Typical System Configuration

Character Class Array

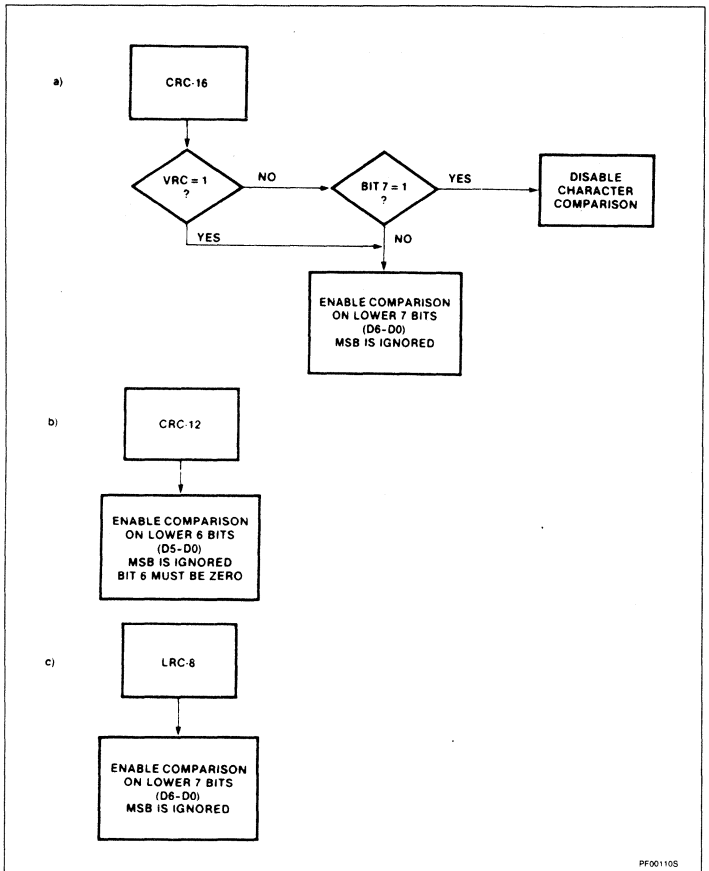
This 128 x 2 array holds the character class associated with each of 128 possible 7-bit characters. The array is zero after a master reset. When the character class array is loaded (see PGC Addressing), the character on the data bus is placed in the class specified by the contents of command register bits CR2 and CR3. The PGC uses these two command bits to represent four different character classes. These are:

1. Normal class (included in the accumulation)
2. SYN character/BISYNC not included class
3. Block terminating character/search class
4. Second search character class (preceded by DLE)

These encoded character classes are used by the PGC:

1. To control the BCC accumulation of associated characters in BISYNC modes only. BCC accumulation in automatic or single accumulation modes is carried out independent of the character classes.
2. To detect characters and two character sequences in all modes of accumulation and to set the control character detect bits in the status register.

It should be noted that any number of characters (up to 128 for CRC-16 or LRC-8; up to 64 for CRC-12) can be put into any one class.



NOTE:
Transparent mode always disables the SSC comparison.

Figure 2. Character Class Array Comparison Operation

If VRC is specified along with CRC-16 then the least significant 7 bits of the character are used for character array comparison. If VRC is not enabled, but CRC-16 is, the MSB of the character then determines whether a character comparison is to take place. If the MSB is 0, the comparison takes place; if the MSB is 1, the comparison does not take place and the character is processed as though it were in the normal class. This enables the PGC to detect all communication control characters and DLE-SSC sequences.

Only the first 64 locations of the array are accessed if CRC-12 is selected. The user should right justify each six bit character (D0-D5) to be written into the character class array. Bit 6 must be zero.

If VRC is enabled, the generated parity becomes the most significant bit of the character to be compared. VRC is not allowed in BISYNC transparent mode.

The method in which the character register contents is compared against the character class array depends on the BCC polynomial chosen. Figure 2 illustrates the comparison process.

DLE Read Only Memory

The DLE characters are stored internally and are selected by the error polynomial as follows:

CRC-12: 01 1111

LCR-8 or CRC-16:
No VRC or odd VRC: 0001 0000
Even VRC: 1001 0000

BCC and Parity Generator

This functional block performs all the necessary computation to generate and update the BCC accumulation on a character by character busis. It contains the three generator polynomials (CRC-16, CRC-12, and LRC-8) that can be selected to compute the BCC. This block also checks and generates odd or even parity for 7-bit (ASCII) characters.

BCC Registers

This block consists of two 8-bit registers (BCC upper and BCC lower) which contain the high and low order bytes of the BCC accumulation. The result of the accumulation from the BCC and parity generator is stored in these registers. A recirculating register address pointer is initialized by a power on, master reset, or clear BCC command. The pointer alternately selects BCC upper and lower on successive BCC register accesses for CRC-16 or CRC-12. For LRC-8, BCC upper is always selected.

BCC upper and lower are cleared by a clear BCC or master reset command. The highest term of the BCC polynomial is always represented by bit 0 of BCC upper; the lowest term is always represented by bit 7 of BCC lower (see figure 3, Orientation of BCC Polynomials.)

The length of the block check character depends on the error checking polynomial that is selected. If LRC-8 is chosen, the BCC result is stored entirely in BCC upper. The BCC lower remains unchanged from previous setting. Both BCC registers are used when CRC-16 is specified. When CRC-12 is selected, the block check character is 12 bits long. The six least significant bits of the BCC are stored in the least significant bits of the BCC lower. The remaining upper six bits of the BCC are stored in least significant bits of BCC upper. The two most significant bits in each BCC register are filled with zero.

The BCC register(s) are read by the CPU after the last data character is transmitted. They can then be sent to the R/T to complete a transmitted block of data. These registers are read and loaded when one PGC is time-shared by several R/Ts. Refer to Applications Information—Multiplexed PGC.

PGC Addressing

All internal registers and the character class array are selected by the unique address codes shown in table 2.

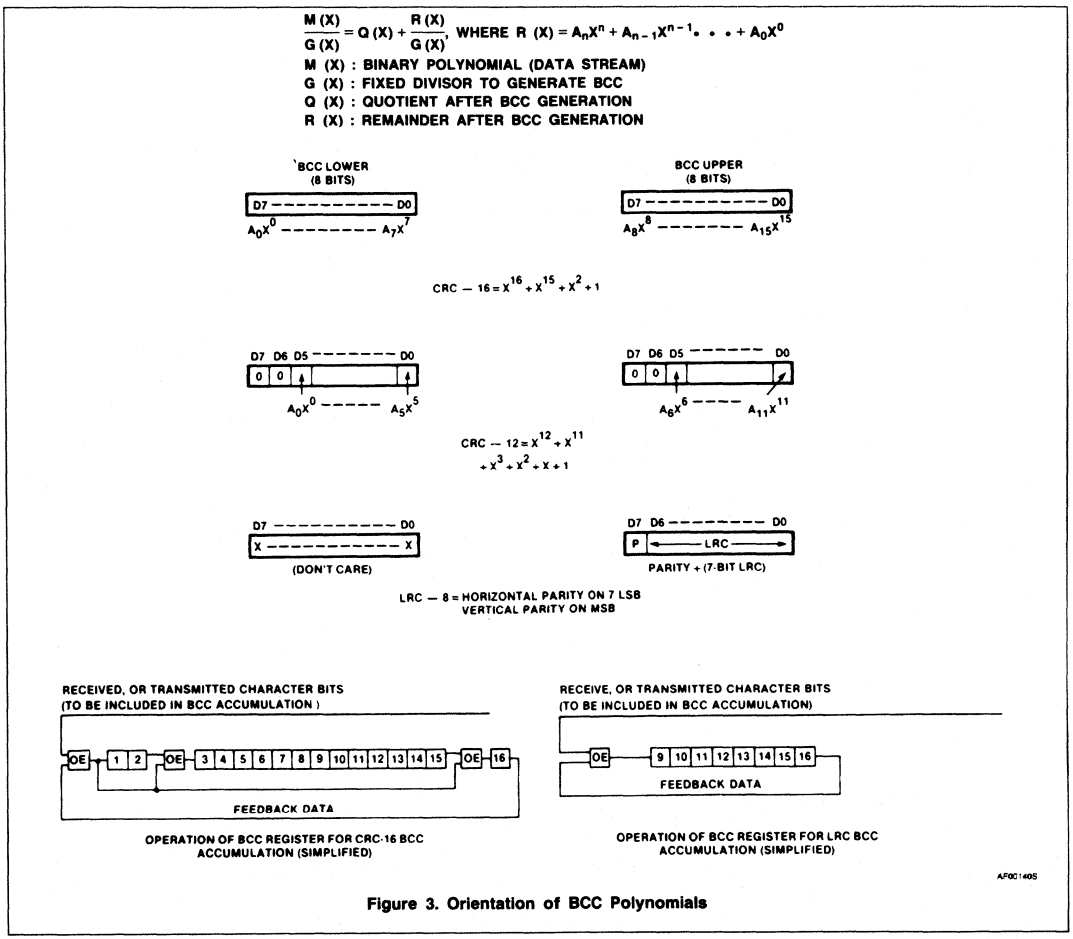


Figure 3. Orientation of BCC Polynomials

Product Specification

Table 2. ADDRESS CODES

$\overline{CE0}$	$\overline{CE1}$	A1	A0	R/$\overline{W}$	FUNCTION
0	0	X	X	X	Operation not guaranteed
0	1	0	0	0	If MR2 = 0 load data bus into character register If MR2 = 1 PGC not selected[1]
0	1	0	0	1	If MR2 = 1 load data bus into character register If MR2 = 0 PGC not selected[1]
0	1	0	1	X	PGC not selected[1]
0	1	1	0	X	PGC not selected[1]
0	1	1	1	X	PGC not selected[1]
1	0	0	0	0	Read character register
1	0	0	0	1	Load data bus into character register if MR1,0 ≠00[2]; write character class array using CR3, CR2 class code if MR 1,0 = 00[3,4]
1	0	0	1	0	Read Status register
1	0	0	1	1	Write command register
1	0	1	0	0	Read mode register
1	0	1	0	1	Write mode register
1	0	1	1	0	Read BCC upper/lower[5]
1	0	1	1	1	Write BCC upper/lower[5]
1	1	X	X	X	PGC not selected[1]

NOTES:
1. Data bus is 3-state
2. Character will not be accumulated unless MR3 = 1.
3. Character will not be accumulated even if MR3 = 1.
4. The mode bits MR1 and MR0 are cleared to 00 by power-on-reset, master reset, or by loading the mode register bits MR1 and MR0.
5. Recirculating internal pointer selects BCC upper on first access, BCC lower on next access for all BCCs except for LRC-8; in case of LRC-8, the pointer only selects BCC upper.

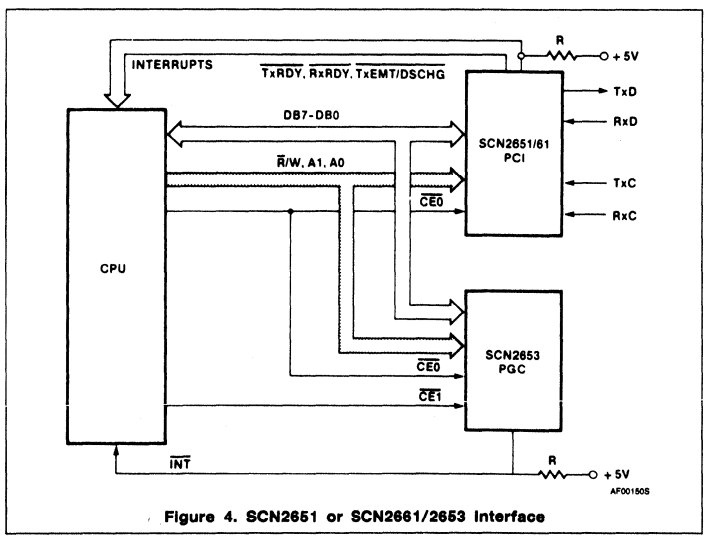


Figure 4. SCN2651 or SCN2661/2653 Interface

INTERFACE SIGNALS AND TIMING

PGC data transfers are controlled by A1, A0, and R/$\overline{W}$ which must be stable prior to the active low going chip enable pulse. $\overline{CE0}$ is used for PGC monitoring of data transfers between a CPU/DMA controller and a R/T; $\overline{CE1}$ is used for direct CPU-to-PGC transfers. MR3 must be set prior to loading the character register in order to accumulate or compare characters via $\overline{CE1}$. The active low (leading) edge of chip enable initiates a PGC read/write cycle; the rising (trailing) edge ends the cycle and also serves as a write strobe.

When loading the character, mode, or command register, the data bus is strobed into the selected register on the trailing (rising) edge of the appropriate $\overline{CE}$. When writing into the character class array, the data on the bus (the special character) is placed in the class specified by command register bits CR3 and CR2.

Characters are transferred into the character register when $\overline{CE0}$ is active (low) depending on the state of MR2 and the R/$\overline{W}$ input. Characters (from the R/T are loaded into the character register when in receive mode (MR2 = 0 and R/$\overline{W}$ = 0) while CPU/DMA characters are loaded into the character register when in transmit mode (MR2 = 1 and R/$\overline{W}$ = 1). The time between consecutive chip enables is given by t_{CEC} or t_{CED}.

The open drain active low interrupt signal ($\overline{INT}$) goes active whenever one or more of four maskable status conditions (SR0-SR3) are true (= 1). A status read deactivates $\overline{INT}$.

The same techniques used in interfacing the SCN2651 PCI to 8-bit microprocessors can be used to interface the SCN2653 PGC (consult Application Note M22). Note that when addressing the R/T's holding registers, the PGC pins must have A1, A0 = 00 and that the address and R/$\overline{W}$ signals must be stable (set up) prior to the active low chip enable. When using the SCN2651 or SCN2661 as the R/T, the PGC's A1, A0, R/$\overline{W}$, and $\overline{CE0}$ are directly connected to comparable SCN2651 or SCN2661 signals. Schematics of an SCN2653 monitoring data transfers to/from the Signetics SCN2651/2661 and SCN2652 are shown in figures 4 and 5.

An alternate interfacing technique is to treat the PGC as an independent peripheral device. This necessitates a write character register instruction after the CPU reads or writes a character to or from the R/T.

Product Specification

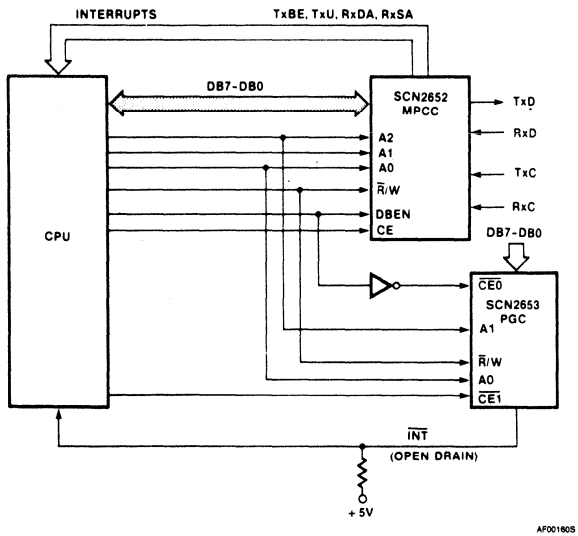


Figure 5. SCN2652/2653 Interface – Typical Protocols: *BISYNC, DDCMP, SDLC, HDLC

PGC PROGRAMMING

The PGC operational mode must be initially programmed by the CPU (see figure 6). The mode register, command register and character class array should be written into, after a power-on-reset or a master reset command. The character class array should be programmed only for the classes pertinent to the application. After a master reset, the character class array is zero which places all characters in the normal class (included in the BCC accumulation).

OPERATION

The PGC should be initially configured by the CPU (via $\overline{CE1}$) prior to systems operation. This is done by loading the mode register, command register and character class array (see PGC PROGRAMMING). Characters may then be loaded into the character register for BCC accumulation, VRC generation/checking, BTC/SC and DLE-SSC comparisons. See table 3 for a summary of BCC accumulation modes.

BCC accumulation depends on the mode selected.

BISYNC Normal

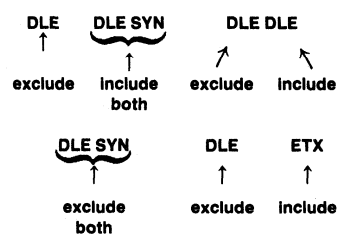
In BISYNC normal mode, all characters loaded into the character register are accumulated except those in the SYN/BISYNC not included class. During receive (MR2 = 0), a BTC/SC match will cause the BCC accumulation to stop after the next one (LRC-8) or two (CRC-12 or CRC-16) characters have been accumulated. At that time, if the BCC accumulation does not equal zero, the BCC error bit (SR0) will be set and $\overline{INT}$ will go active if the corresponding mask bit (CR4) is enabled (= 1). In transmit (MR2 = 1), the BCC accumulation is automatically stopped once the BTC/SC character has been accumulated. The CPU must read the BCC upper and BCC lower (CRC-12 or CRC-16) register(s) and transmit them to the R/T.

Note that the received BCCs are not subject to VRC if CRC-16 is selected. If LRC-8 is selected, the received BCC is subject to VRC. An incorrect result will set the VRC error bit (SR1). After its accumulation, the least significant 7 bits of BCC upper are checked and a non-zero result will set the BCC error bit (SR0). BCCs are not checked against the character class array nor are they compared to the DLE ROM.

Second search character (SSC) detection is enabled so that a DLE-STX or two character communication control sequence can be detected.

BISYNC Transparent

BISYNC transparent mode should be used for data blocks beginning with DLE-STX if the DLEs are transferred between CPU and R/T ($\overline{CE0}$) or CPU and PGC ($\overline{CE1}$), i.e., DLEs are not stripped. VRC should be disabled in this mode. Characters excluded from the BCC accumulation are the first DLE of a DLE-non SYN sequence pair and the DLE-SYN sequence if not preceded by an odd number of DLEs. For example, consider the following transparent mode character string:



Product Specification

Table 3. SUMMARY OF BCC ACCUMULATION MODES

ACCUMULATION MODES	START ACCUMULATION	STOP ACCUMULATION	CHARACTERS EXCLUDED FROM ACCUMULATION
BISYNC normal and BISYNC transparent	Clear BCC registers command Mode register is loaded with BISYNC or automatic mode Start accumulation command Load BCC registers	After BTC has been detected and received BCC is accumulated After transmitted BTC has been accumulated Single mode is selected	SYN/BISYNC not included class in normal mode DLE-SYN/not included class and first DLE of a DLE non SYN pair in transparent mode. These characters are not excluded if preceded by an odd number of DLEs
Automatic	Same as above	Single mode selected	None
Single	Start accumulation command	After each character has been accumulated	Up to user who must generate start accumulation command for each character to be included

In receive and transmit modes, the termination of BCC accumulation works exactly as in BISYNC normal, except that the BTC/SC must be immediately preceded by an odd number of DLEs to be identified as a BTC/SC.

Second search character detection is not enabled in BISYNC transparent.

After a BTC/SC class character is detected by the PGC when receiving in either BISYNC mode, the following one or two characters are accumulated (depending on LRC-8 or CRC-12/16, respectively) and the PGC will automatically stop further accumulation. However, the PGC can continue the accumulation if a start accumulate command is issued or either BISYNC mode is loaded into the mode register. The start accumulate command should be given to the PGC before loading the character that follows the detected BTC/SC. This procedure enables a special search character to be detected (the BTC/SC detect bit (SR2) will be set and an interrupt generated if CR6 = 1) with the BCC accumulation continuing (see figures 7 and 8).

Automatic Accumulate

All characters loaded into the character register are accumulated, BTC/SC and SSC detection is enabled. The BCC accumulation is not automatically terminated. (The CPU must use single accumulate mode to stop the accumulation). When in receive mode, the BCC error bit (SR0) is set/reset after accumulating each character so that the CPU must examine this bit after the last character is accumulated. SR0 = 0 if the accumulated remainder in the BCC register(s) is zero; otherwise SR0 = 1. Examples of use of automatic accumulate mode usage include an R/T (SCN2651/SCN2661) in transparent DLE/SYN strip mode and asynchronous/synchronous/parallel DDCMP.

Single Accumulate

All characters for which a start accumulate command (CR1, CR0 = 01) is given are accumulated and compared against the character class array. If not given, the BCC accumulation is not updated and BTC/SC and SCC detection is disabled. Operation in this mode is otherwise identical to automatic accumulate.

Single accumulate mode can be used to selectively accumulate characters under CPU control or to accumulate characters that were unintentionally excluded in one of the other modes.

Polynomial Selection and DLE Comparison

The BCC polynomial may be CRC-16, CRC-12 or LRC-8. The cyclic redundancy check (CRC) is generated by dividing the binary value of a character in the character register by the selected polynomial. The quotient is discarded and the remainder is used as the BCC (two 6-bit characters for CRC-12, two 8-bit characters for CRC-16). CRC-16 uses all 8 bits of each BCC register. CRC-12 uses the least significant 6 bits of the BCC registers. The two most significant bits of the BCC registers are cleared to zero whenever CRC-12 is selected (see figure 3).

When the PGC is in receive mode (MR2 = 0), the received BCC will be accumulated. The result will be zero for an error free message.

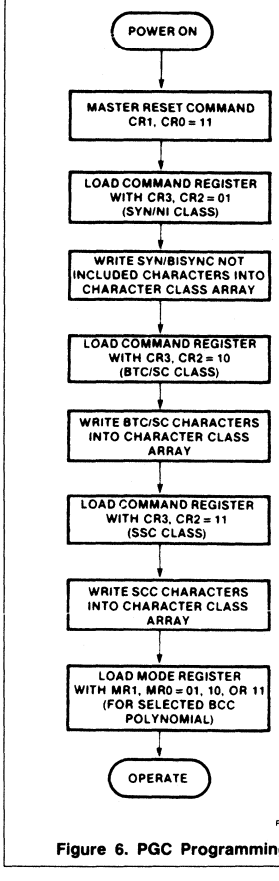
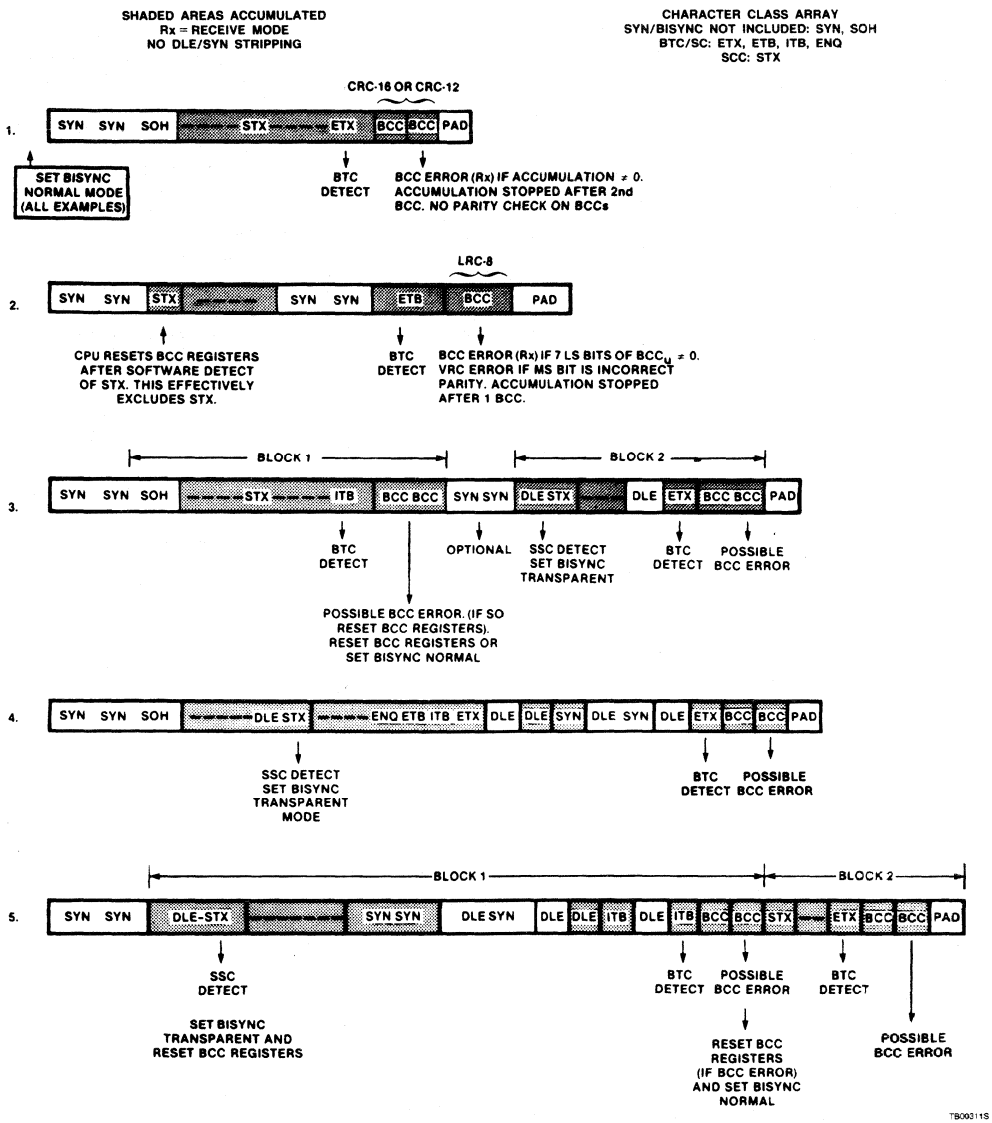


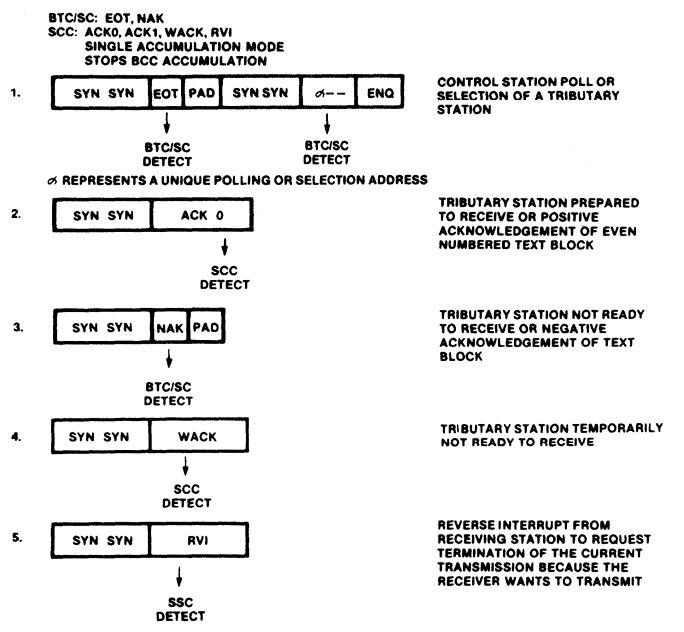
Figure 6. PGC Programming



NOTES:

1. BCC error only for receive mode. In transmit mode, CPU must respond to BTC detect by reading the BCC register(s) and sending them to the R/T. The accumulation is stopped after the BTC is accumulated.
2. ENQ (DLE – ENQ) in a text message should be treated as an abort.
3. Opening SYNs may be stripped by the R/T.
4. The single accumulate mode and command can be used to accumulate a character that inadvertently was excluded. (For example, the DLE of a DLE – STX if the PGC was in transparent mode and there was not a line turnaround prior to the DLE). The single accumulation should be done using $\overline{CET}$ after the BCC(s) have been accumulated.

Figure 7. Examples – Bisync Text Message BCC Accumulation



NOTES:
1. BCC accumulation should be ignored for control messages. This can be effected by single accumulate mode without single accumulate commands.
2. Characters programmed as SSCs should be the binary equivalent of the second character of the DLE – SSC sequence.

Figure 8. Examples – Bisync Control Messages

CRC-12 is used with 6-bit codes. The internal 6-bit transcode DLE character hex 1F is selected by CRC-12. VRC should be disabled (MR4 = 0) for CRC-12 operation. The two most significant bits of the character register are ignored when compared to the internal 6-bit DLE. When the character is checked against the character class array, the MSB is ignored and the next MSB (bit 6) is assumed to be zero. If CRC-12 is specified, the user must write to the character class array with bit 6 cleared.

CRC-16 or LRC-8 implies the use of ASCII or EBCDIC although any 7-bit plus parity or 8-bit no parity code may be used (with DLE = hex 10 or hex 90). The DLE character compare is on an 8-bit basis with the generated parity (if VRC is enabled) as the MSB. When the character is compared against the character class array, the MSB is not used. This may result in a false BTC or SSC detection if there is a VRC error. However, the VRC error bit (SR1) will be set under that condition.

The LRC-8 is generated by the exclusive OR of the 7 least significant bits of the character register and the BCC upper. The most significant bit of the LRC-8 check character is a vertical odd/even parity bit (MR5 = 0/1), which is generated on the least significant bits of that character. The selection of LRC-8 implies VRC is enabled and that only the BCC upper is used for the BCC accumulation. The BCC lower remains unchanged from previous setting.

VRC Generation and Detection

Parity (VRC) is enabled by MR4 and specified as odd or even by MR5. VRC should be disabled when in BISYNC transparent mode and whenever CRC-12 or CRC-16 (EBCDIC) is selected as the BCC polynomial. MR4 = 1 enables VRC generation and detection for both receive and transmit operations. Characters loaded into the character register will have VRC generated on the least significant 7 bits with the generated parity bit written into the character register MSB. If the generated parity does not match the MSB of the loaded character, the VRC error bit (SR1) is set and $\overline{INT}$ asserted if the corresponding mask bit was enabled (CR5 = 1). Thus, if 7-bit characters are to be transmitted with VRC, CR5 should be zero and SR1 ignored. 8-bit characters with a VRC bit in the MSB position are parity checked by the PGC in both transmit (to R/T) and receive (from R/T) modes, i.e., the PGC operates as a data bus parity checker.

CHARACTER CLASSES

Normal (Included in the Accumulation)

Any character that belongs to this class is normal data, i.e., the character is not a communication control or other special character. Characters in this class are always accumulated in BISYNC, automatic and single accumulation modes.

SYN Character/BISYNC Not Included

SYN characters are never accumulated in BISYNC normal accumulation mode. In BISYNC transparent accumulation mode, the DLE-SYN character pair is not accumulated, but a SYN not preceded by a DLE is accumulated. (DLE is implied as an odd number of DLEs).

Block Termination Character (BTC)/Search Character (SC)

BTC/SC characters have two functions in the PGC: termination of BCC accumulation and character detection. In BISYNC transparent mode, a BTC/SC must be preceded by an odd number of DLEs to be recognized.

Termination of BCC Accumulation

In BISYNC normal and transparent accumulation modes, the PGC will stop the accumulation upon the detection of the BTC/SC character. Examples of BTCs are ETX, ETB, ITB, ENQ.

In receive mode, the accumulation is stopped after the following one (LRC-8) or two (CRC-12, CRC-16) character(s) have been accumulated. In transmit mode, the accumulation is stopped after the BTC/SC character has been accumulated. The BTC/SC character is always accumulated in all of the accumulation modes.

Character Detection

BTC/SC characters will be detected in any of the four accumulation modes when that character is being accumulated. The BTC/SC status bit (SR2) is set on detection. Since detection also stops BISYNC BCC accumulation, the BISYNC accumulation must be restarted if the character is not a BTC. This can be effected by loading BISYNC mode into the mode register or generating a start accumulation command.

Product Specification

Second Search Character Class (SSC)

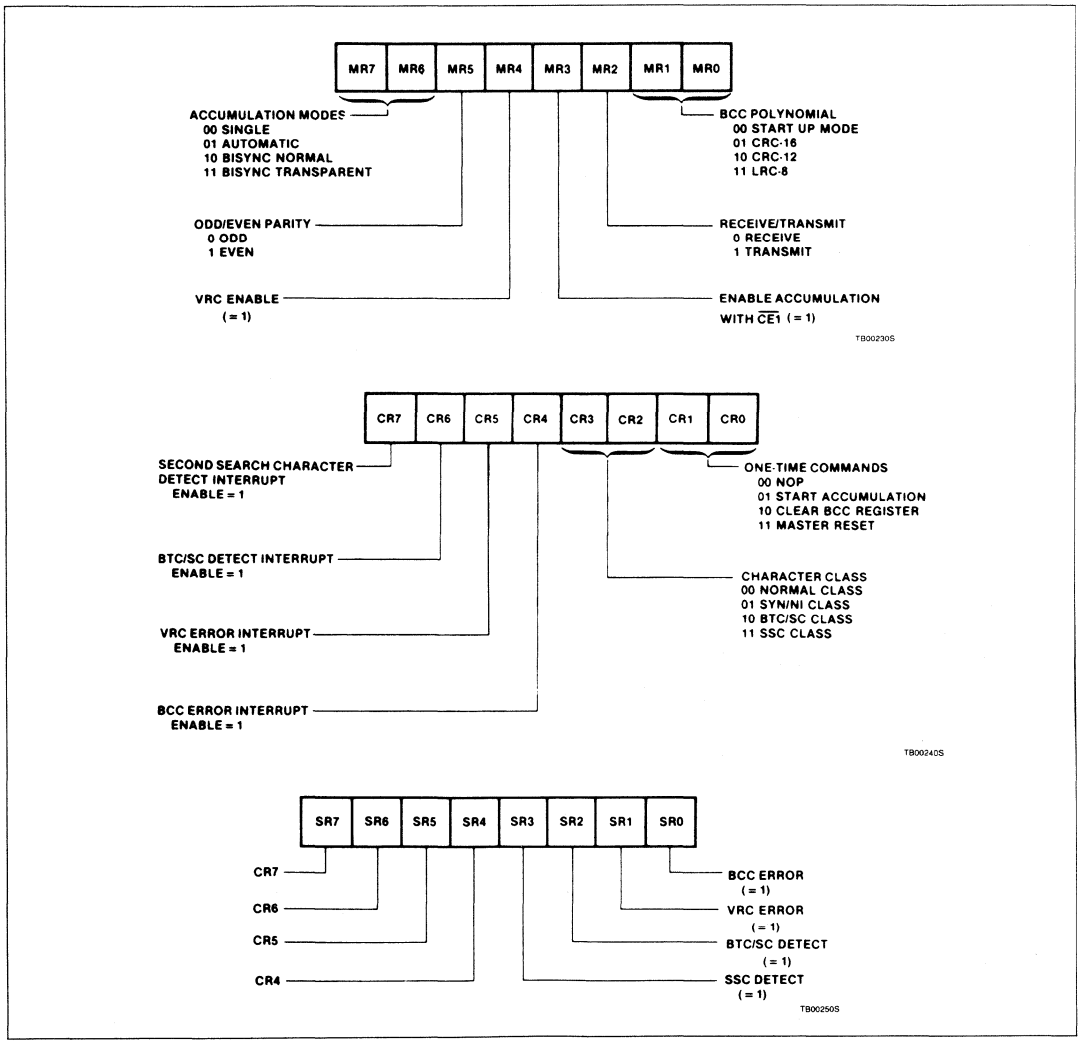
Control functions in character oriented data link control procedures can be represented by a sequence of two characters, the first character being a DLE. Examples include ACK0, ACK1, WACK, RVI, DISC, WBT and the initiation of transparent text (DLE,STX). The PGC will detect such sequences, except in BISYNC transparent mode, when an SSC class character is being accumulated after being immediately preceded by an odd number of DLEs. Under those conditions, the SSC status bit (SR3) will be set.

The SSC character is always accumulated in all of the accumulation modes.

REGISTER BIT DESCRIPTION

The operation of the PGC is determined by programming the mode register and the command register. The status register provides feedback on potential interrupt conditions. Formats of these registers are shown in table 4.

Table 4. PGC REGISTER BIT FORMATS



Product Specification

Table 5. BCC ACCUMULATION BY CHARACTER CLASS

CR3	CR2	CLASS	BISYNC NORMAL	BISYNC TRANSPARENT	AUTOMATIC ACCUM	SINGLE ACCUM
0	0	Normal	Yes	Yes	Yes	Yes
0	1	SYN/BISYNC not included	No	Yes, unless preceded by an odd number of DLEs	Yes	Yes
1	0	BTC/SC	Yes	Yes	Yes	Yes
1	1	SSC*	Yes	Yes	Yes	Yes

NOTE:
*Preceded by DLE

Mode Register

The mode register defines general PGC operation characteristics. MR1 and MR0 = 00 permit the character class array to be programmed. These bits will be zero after a power on or master reset command. After the character class array is programmed, these bits should be set to 01,10, or 11 to select the CRC-16, CRC-12 or LRC-8 polynomials.

MR2(Tx/$\overline{Rx}$) determines whether or not the PGC is to generate (Tx) or generate and check (Rx) the BCC. It is used with $\overline{R}$/W to determine if the data bus is to be loaded into the character register when $\overline{CE0}$, $\overline{CE1}$, A1, A0 = 0100.

If MR2 = 1: 1) the PGC will generate the BCC but will never set the BCC error bit (SR0). 2) If the $\overline{R}$/W pin is high when $\overline{CE0}$, $\overline{CE1}$, A1, A0 = 0100, then the data bus will be loaded into the character register. If $\overline{R}$/W is low under these conditions, the PGC is not selected.

If MR2 = 0: 1) the PGC will accumulate the BCC and set the BCC error bit (SR0) when appropriate. 2) If the $\overline{R}$/W pin is low when $\overline{CE0}$, $\overline{CE1}$, A1, A0 = 0100, then the data bus will be loaded into the character register. If $\overline{R}$/W is high under these conditions, the PGC is not selected.

MR3 is a $\overline{CE1}$ accumulate/compare enable bit. If MR3 = 0, characters loaded into the character register by $\overline{CE1}$ are not accumulated, checked against the character class array, or compared to the DLE ROM. Parity will be generated and checked if VRC is enabled (MR4 = 1). The primary use of MR3 = 0 is to generate parity on a 7-bit character which is to be transmitted to an R/T. The CPU loads the character register with the 7-bit character and reads the 8-bit VRC generated character via $\overline{CE1}$. This 8-bit character is then transferred to the R/T via $\overline{CE0}$. Another application of MR3 = 0 is for a CPU to interleave parity checking on memory data ($\overline{CE1}$) with on line R/T data transfer ($\overline{CE0}$).

If MR3 = 1, characters loaded into the character register by $\overline{CE1}$ will be accumulated (according to the BCC accumulation mode selected) and compared against the character class array and DLE ROM. This bit setting should be used when the CPU/DMA controller sends data characters to be accumulated or compared to the PGC and the R/T is inactive (off line). If the R/T were active, then a DLE or BTC loaded into the character register via $\overline{CE0}$ would cause incorrect accumulation and character comparisons if the next character was loaded via $\overline{CE1}$.

Table 6. BTC/SC AND SSC DETECTION CONDITIONS

CLASS	BISYNC NORMAL	BISYNC TRANSPARENT	AUTO ACCUM	SINGLE ACCUM
BTC/SC	Yes*	Yes*	Yes*	Yes* †
SSC	Yes*	No	Yes*	Yes* †

NOTES:
* Only if immediately preceded by an odd number of DLEs.
† Start accumulate command necessary for detection.

MR4 is a VRC enable bit. If MR4 = 1, VRC is enabled as odd/even by MR5. VRC is generated on the 7 LS bits of the character and the MS bit is checked against the generated parity. If not equal, SR1 is set. If MR4 = 0, VRC is not enabled. MR4 = 0 is used for BISYNC transparent mode with ASCII code, and for both BISYNC modes for EBCDIC and SBT.

MR5 is an odd/even VRC bit. If MR5 = 1, the total number of 1 bits in the character including the parity bit is even. If MR5 = 0, the total number of bits is odd. This bit is ignored if MR4 = 0.

MR7, MR6 select the BCC accumulation mode. These modes have been previously discussed in the operation section.

Command Register

The command register contains four interrupt enables, a 2-bit character class code used when programming the character class array, and 2 bits that specify three one time commands and a NOP.

CR1, CR0 = 00 is a NOP. This bit setting is used when changing CR7-CR2 without affecting any of the 3 one time commands.

CR1, CR0 = 01 is a start BCC accumulation command. In single accumulation mode, the character accumulated is the character that is in the character register at the time the command is given. The accumulation stops immediately after the character has been accumulated. If the command is given in either of the BISYNC or automatic accumulation modes, it enables the PGC to accumulate the BCC starting with the next character loaded into the character register. This is a means of restarting a BISYNC normal accumulation after detection of a BTC/SC that is not a valid BTC (example; CR, LF, TAB). In all accumulation modes, a previously detected DLE will not be cancelled by this command.

CR1, CR0 = 10 is a clear BCC registers command. Both BCC registers are cleared along with the associated internal pointer and SR0-SR3. The pointer points to BCC upper. $\overline{INT}$ is forced high. This command permits BCC accumulation, starting with the next character loaded into the character register in BISYNC or auto modes. Single accumulate mode requires a start BCC accumulation command.

CR1, CR0 = 11 is a master reset command. All internal registers (except the character register), the internal pointer, and the entire character class array are cleared. $\overline{INT}$ is forced high.

CR3 and CR2 are used for programming the character class array. During a write character class array instruction, the character corresponding to the 7 LS bits of the data bus is placed in the class contained in CR3 and CR2. The encoded character classes control the accumulation of the associated character as shown in table 5.

Detection operates under the conditions shown in table 6.

CR7, CR6, CR5, CR4 are interrupt enables that individually enable/disable $\overline{INT}$ when the corresponding status register condition is true (set). Each bit is set in order to enable $\overline{INT}$ upon the condition. Each bit is reset to disable $\overline{INT}$ upon the condition. The state of

these bits may be read via the status register (SR7, SR6, SR5, SR4).

The corresponding status bits (SR3, SR2, SR1, SR0) are set independent of the interrupt enables. The bit assignments are:

CR4–BCC error interrupt enable

CR5–VRC error interrupt enable

CR6–BTC/SC detect interrupt enable

CR7–DLE-SSC detect interrupt enable

Status Register

This register reflects the status of the 4 conditions that are potential interrupt ($\overline{INT}$) sources and the 4 interrupt enables in the command register. A status register read clears SR0, SR1, SR2, SR3 and deactivates $\overline{INT}$. These bits are also cleared by a master reset or clear BCC command.

SR0 is a BCC error bit. This bit can only be set in receive mode (MR2 = 0). In BISYNC normal and BISYNC transparent modes, SR0 will be set/reset once the accumulation has been stopped by the detection of the BTC/SC character and accumulation of the BCC(s).

In automatic and single accumulate modes, SR0 is set/reset after each character in the character register has been accumulated.

The rules for the detection of a BCC error are:

SR0 = 1 LRC-8: 7 least significant bits of BCC upper $\neq 0$
CRC-12, CRC-16: BCC upper or BCC lower $\neq 0$

SR0 = 0 LRC-8: 7 least significant bits of BCC upper = 0
CRC-12, CRC-16: BCC upper and BCC lower = 0

SR1 is a VRC error bit. When set, this bit reports a character parity error (on receive or transmit) when parity is enabled (MR4 = 1). Parity is odd/even as specified by MR5. The parity bit will be regenerated in the character register.

SR2 is a BTC/SC detect bit. When set, this bit indicates the character being accumulated is of the BTC/SC class for BISYNC normal, automatic and single accumulate modes. In BISYNC transparent mode, the BTC/SC character being accumulated must be immediately preceded by an odd number of DLEs for this bit to be set.

SR3 is a DLE SSC detect bit. This bit can only be set when in BISYNC normal, auto, or single accumulated modes. When set, it indicates that the character being accumulated is of the SSC class when that character was immediately preceded by an odd number of DLEs.

SR7, SR6, SR5, SR4 are interrupt enables. These 4 bits reflect the state of the interrupt enable command bits CR7, CR6, CR5, and CR4, as follows:

SR4–BCC error

SR5–VRC error

SR6–BTC/SC detect

SR7–SSC detect

APPLICATIONS INFORMATION

Dedicated PGC

The most efficient use of the 2653 is to dedicate one to each R/T for two way alternate (half duplex) operation or two to each R/T for two way simultaneous (full duplex) operation (see figure 9). The CPU configures each PGC (using $\overline{CE1}$) by initializing the the mode register, command register, and character class array. Data transfers to or from the R/T can then be on a DMA basis with each receiver holding register ready signal used as a read request (RREQ) and each transmit holding register available signal used as a write request (WREQ) to the DMA controller. The CPU needs only to respond to enable interrupts from each of the PGCs. The individual $\overline{INT}$ outputs can be wire-OR'd into single CPU interrupt ($\overline{INTRPT}$) with one pull up resistor. Each PGC in this system has a unique address that is decoded into the respective chip enables.

The CPU or DMA controller could send a block of memory data to the PGC to be error checked without sending that data to the R/T. In that case, $\overline{CE1}$ is used.

Multiplexed PGC

One PGC may be time-shared among a few R/Ts if the CPU saves and restores the mode register and partial BCC result in the BCC registers. These registers are accessed via $\overline{CE1}$. There must be a separate save area for each R/T (serial channel) and a channel pointer indicating the last R/T that transferred or received a data character (see figure 10).

The loading of the BCC registers will clear SR0-SR3 and all previously detected special characters, i.e., DLE, BTC/SC, BCC (BISYNC modes). The BCC accumulation will start again when the next character is loaded into the character register in all accumulation modes except single. That mode requires a start accumulation command.

Figures 11 and 12 represent software flow diagrams for transmit and receive service requests. Note that interrupts from all other R/Ts must be masked during a read or write to the BCC registers so as not to affect the internal BCC address pointer. It is recommended that all R/T interrupts be masked while servicing an interrupt that accesses any PGC register.

BISYNC Operation

Table 7 is a concise listing of SCN2651/SCN2661 operating modes with recommended corresponding SCN2653 BCC accumulation modes.

Character Comparator

The PGC can be used as a programmable data bus character comparator which monitors data bus transfers (CPU ↔ peripheral, CPU ↔ CPU, CPU ↔ memory, memory ↔ peripheral (via DMA)). The user selectively loads the character class array with BTC/SC and SSC characters to be compared. Status bits will be set and an interrupt can be generated upon SC and DLE–SSC detection. A match on one to 128 different characters or DLE–SSC sequences can be programmed.

Figure 13 depicts an arrangement where the DMA controller or slave CPU handles data bus transfers, the PGC interrogates the data bus, and the host CPU responds to PGC interrupts.

Product Specification

Table 7. BISYNC (ANSI 3.28, ISO 1745) MODES FOR SCN2651/SCN2661 AND SCN2653

SCN2651/SCN2661 OPERATING MODES	SCN2653 BCC ACCUMULATION MODE
Sync normal non-strip	BISYNC normal
Sync transparent non-strip	BISYNC transparent
Normal SYN/DLE strip[1]	BISYNC normal
Transparent SYN/DLE strip[1]	Automatic accumulate[2]
Async (with SYN/DLE characters)	BISYNC normal

NOTES:
1. CPU should switch to non-strip mode after BTC detect. Otherwise a received BCC could be inadvertently stripped.
2. SSC detect should be ignored.

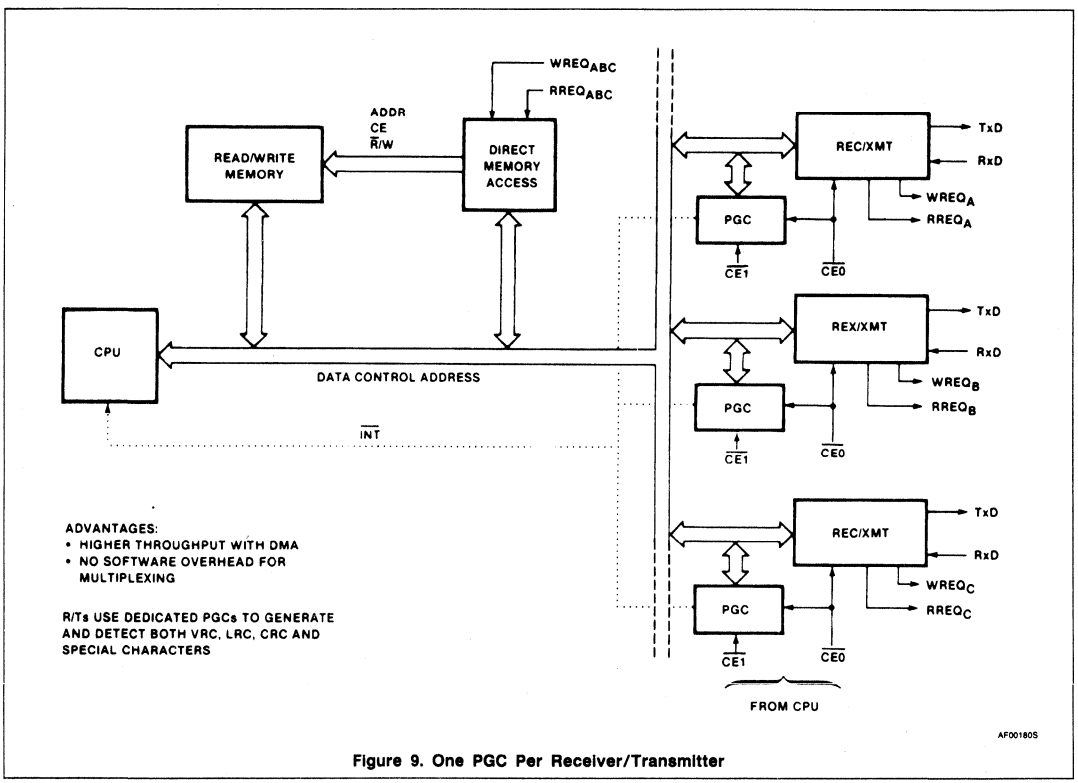


Figure 9. One PGC Per Receiver/Transmitter

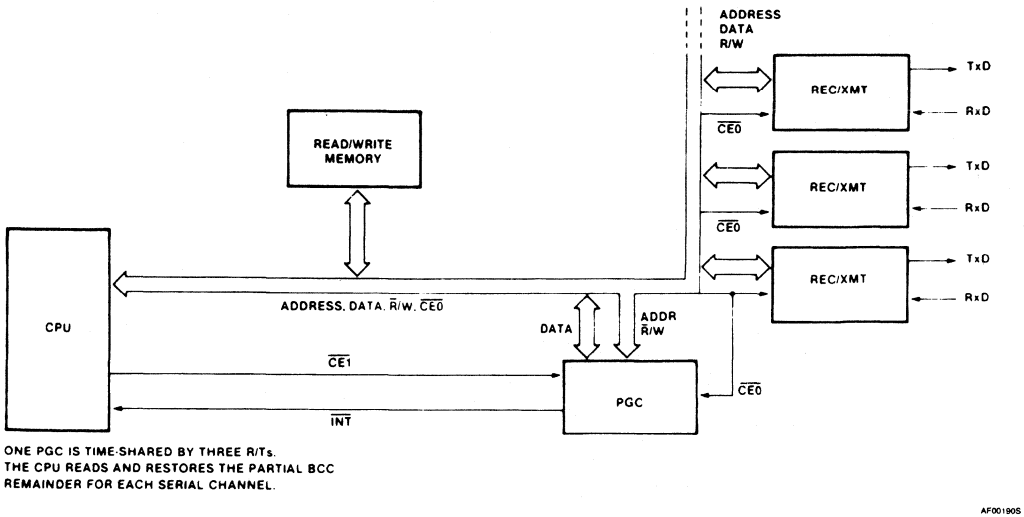


Figure 10. PGC Services Multiple Receivers/Transmitters

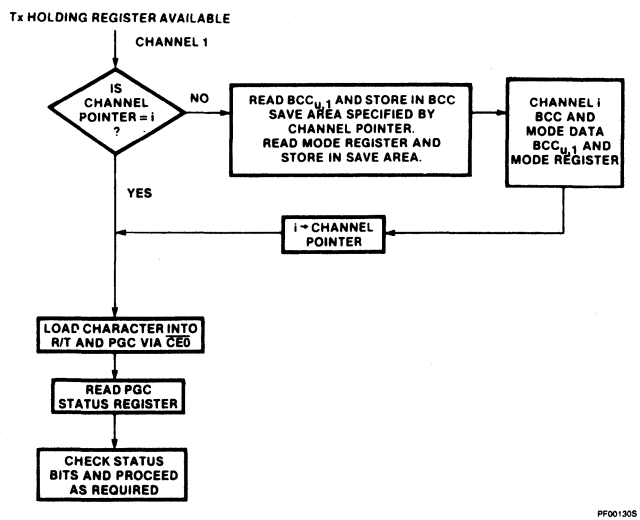


Figure 11. Multiplexed PGC—Transmit to R/T

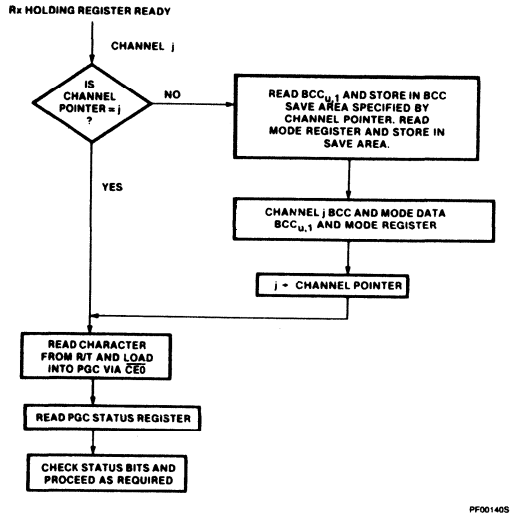
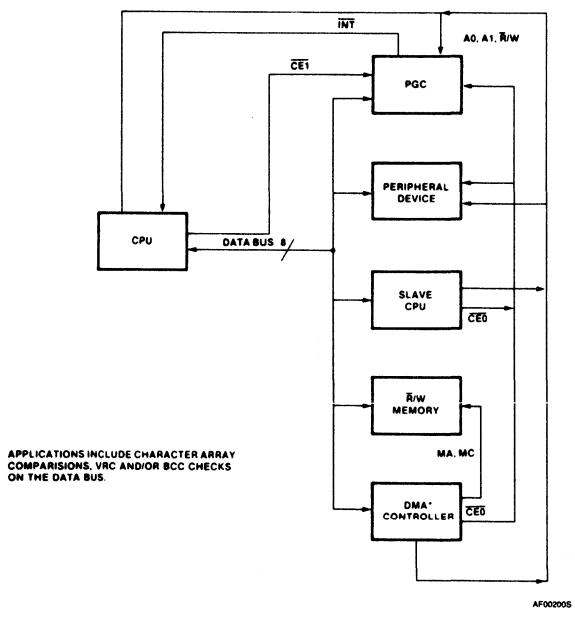


Figure 12. Multiplexed PGC—Receiver from R/T



NOTE:
* CPU initializes DMA controller for each block transfer of data.

Figure 13. PGC Data Bus Monitoring with DMA Transfers

Product Specification

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation sections of this specification is not implied.
2. For operating at elevated temperatures the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. However, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.0V± 5%

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	I_{OL} = 2.2mA		0.25	0.45	
V_{OH}	High	I_{OH} = −400μA	2.4	2.8		
I_{IL}	Input load current	V_{IN} = 0 to 5.5V			10	μA
Output leakage current						μA
I_{LD}	Data bus	V_{OUT} = 4.0V			10	
I_{LO}	Open drain	V_{OUT} = 4.0V			10	
I_{CC}	Power supply current			45	75	mA

AC ELECTRICAL CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.0V± 5%[1, 2, 3]

PARAMETER		LIMITS		UNIT
		Min	Max	
t_{CE}	Chip enable pulse width	250		ns
t_{CED}	Chip enable period D	1750		ns
t_{CEC}[4]	Chip enable period C	1750		ns
t_{AS}	Address set-up	10		ns
t_{AH}	Address hold	10		ns
t_{CS}	Control set-up	10		ns
t_{CH}	Control hold	10		ns
t_{DS}[5]	Data set-up	150		ns
t_{DH}	Data hold	15		ns
t_{DD}[6]	Data delay time for read		200	ns
t_{DF}[6]	Data bus floating time for read		100	ns
t_{INTL}[7]	Interrupt low delay		1600	ns
t_{INTH}[7]	Interrupt high delay		600	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. All voltage measurements are referenced to ground. All time measurements are at 50% level for inputs and at the 0.8V or 2.0V level for outputs. Input levels for testing are 0.45V and 2.4V.
3. Typical values are at +25°C, typical supply voltages and typical processing parameters.
4. t_{CEC} = 600ns during PGC initialization when no BCC accumulation is in progress.
5. t_{DS} = 50ns whenever $\overline{CE0}$ is used.
6. Test conditions: C_L = 150pF.
7. $\overline{INT}$ is an open drain output.

Product Specification

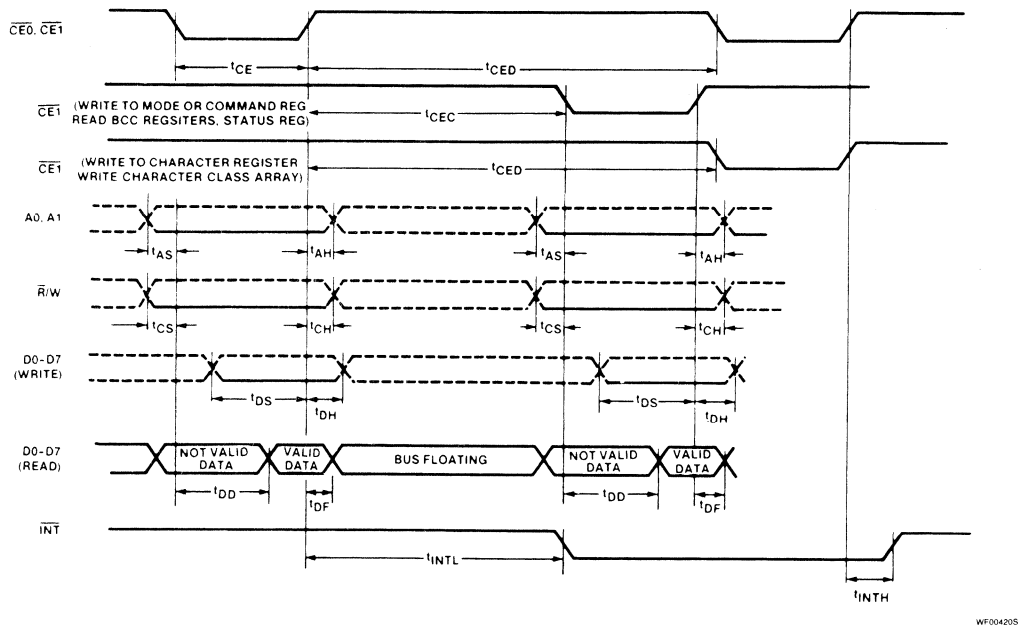


Figure 14. PGC Timing

ENHANCED PROGRAMMABLE COMMUNICATIONS INTERFACE

Originally published by Signetics May 1983

DESCRIPTION

The Signetics SCN2661 EPCI is a universal synchronous/asynchronous data communications controller chip that is an enhanced version of the SCN2651. It interfaces easily to all 8-bit and 16-bit microprocessors and may be used in a polled or interrupt driven system environment. The SCN2661 accepts programmed instructions from the microprocessor while supporting many serial data communications disciplines—synchronous and asynchronous—in the full or half-duplex mode. Special support for BISYNC is provided.

The EPCI serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The SCN2661 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode. Each version of the EPCI (A, B, C) has a different set of baud rates.

The EPCI is available in two packages: a 28-pin (0.6" wide) DIP and a 24-pin (0.4" wide) DIP. The following are the differences between the 24-pin and the 28-pin versions:

1. The 24-pin version provides a single interrupt output ($\overline{INTR}$) instead of the three interrupt outputs ($\overline{RxRDY}$, $\overline{TxRDY}$, $\overline{TxEMT/DSCHG}$) supplied on the 28-pin version. $\overline{INTR}$ will be asserted (low) when one or more of the status bits SR0, SR1 or SR2 is a logic one.
2. Two modem interface pins, the $\overline{DTR}$ output and the $\overline{DSR}$ input, are eliminated in the 24-pin version. Because of this, status bit SR7 should be ignored and the setting of status bit SR2 due to a data set change (DSCHG) can be caused only by a change of the $\overline{DCD}$ input. Since the $\overline{DTR}$ output is eliminated, command register bit CR1 does not perform any function, although it remains writable and readable.

Other than the above, the functional operation, DC electrical characteristics, and AC electrical characteristics of the 24-pin version are identical to the 28-pin version.

FEATURES

- **Synchronous operation**
  - **5 to 8-bit characters plus parity**
  - **Single or double SYN operation**
  - **Internal or external character synchronization**
  - **Transparent or non-transparent mode**
  - **Transparent mode DLE stuffing (Tx) and detection (Rx)**
  - **Automatic SYN or DLE-SYN insertion**
  - **SYN, DLE and DLE-SYN stripping**
  - **Odd, even, or no parity**
  - **Local or remote maintenance loop back mode**
  - **Baud rate: dc to 1M bps (1X clock)**
- **Asynchronous operation**
  - **5 to 8-bit characters plus parity**
  - **1, 1½ or 2 stop bits transmitted**
  - **Odd, even, or no parity**
  - **Parity, overrun and framing error detection**
  - **Line break detection and generation**
  - **False start bit detection**
  - **Automatic serial echo mode (echoplex)**
  - **Local or remote maintenance loop back mode**
  - **Baud rate: dc to 1M bps (1X clock)**
    dc to 62.5K bps (16X clock)
    dc to 15.625K bps (64X clock)

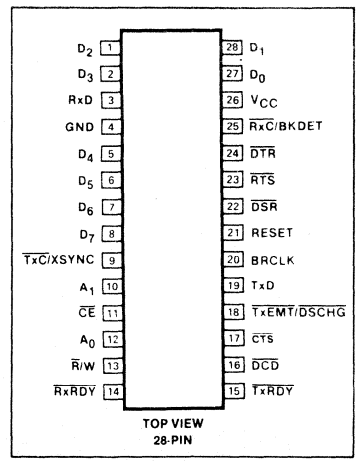
OTHER FEATURES

- **Internal or external baud rate clock**
- **3 baud rate sets**
- **16 internal rates for each set**
- **Double buffered transmitter and receiver**
- **Dynamic character length switching**
- **Full or half duplex operation**
- **TTL compatible inputs and outputs**
- **RxC and TxC pins are short circuit protected**
- **Single 5V power supply**
- **No system clock required**

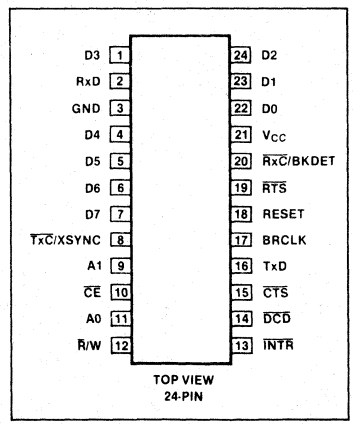
APPLICATIONS

- **Intelligent terminals**
- **Network processors**
- **Front end processors**
- **Remote data concentrators**
- **Computer to computer links**
- **Serial peripherals**
- **BISYNC adaptors**

PIN CONFIGURATIONS



TOP VIEW
28-PIN



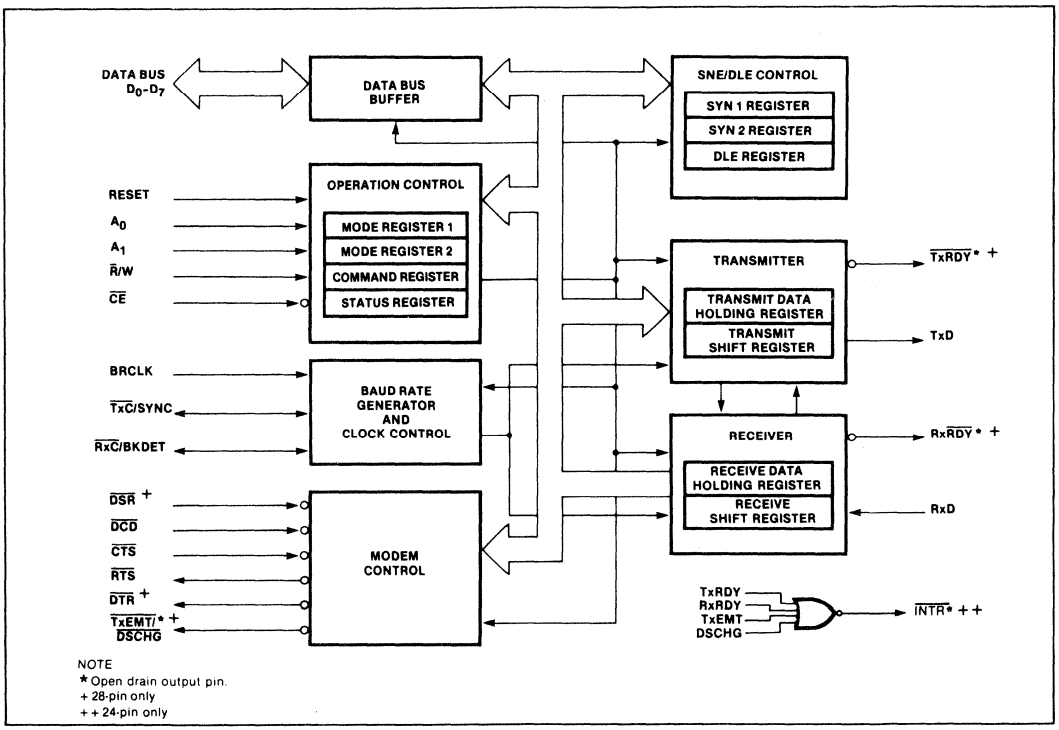
TOP VIEW
24-PIN

ORDERING CODE

	$V_{CC} = 5V \pm 5\%$		
	COMMERCIAL	AUTOMOTIVE	MILITARY
PACKAGES	0°C to +70°C	−40°C to +85°C	−55°C to +125°C
Ceramic DIP 28-Pin 0.6" Wide	SCN2661AC1I28 SCN2661BC1I28 SCN2661CC1I28	SCN2661AA1I28 SCN2661BA1I28 SCN2661CA1I28	SCN2661AM1I28 SCN2661BM1I28 SCN2661CM1I28
Plastic DIP 28-Pin 0.6" Wide	SCN2661AC1N28 SCN2661BC1N28 SCN2661CC1N28	Contact Factory	Not Available
Plastic DIP 24-Pin 0.4" Wide	SCN2661AC1N24 SCN2661BC1N24 SCN2661CC1N24	Contact Factory	Not Available

NOTES
1. See table 1 for baud rates. Specify SCN2661A, B, or C depending on baud rate selected.
2. The SCN68661 is identical to the SCN2661. Order using part numbers above.

BLOCK DIAGRAM



BLOCK DIAGRAM

The EPCI consists of six major sections. These are the transmitter, receiver, timing, operation control, modem control and SYN/DLE control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers 1 and 2, the command register, and the status register. Details of register addressing and protocol are presented in the EPCI programming section of this data sheet.

Table 1 BAUD RATE GENERATOR CHARACTERISTICS SCN2661A (BRCLK = 4.9152MHz)

MR23-20 / BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000 — 50	0.8kHz	-	6144
0001 — 75	1.2	-	4096
0010 — 110	1.7598	−0.01	2793
0011 — 134.5	2.152	-	2284
0100 — 150	2.4	-	2048
0101 — 200	3.2	-	1536
0110 — 300	4.8	-	1024
0111 — 600	9.6	-	512
1000 — 1050	16.8329	0.196	292
1001 — 1200	19.2	-	256
1010 — 1800	28.7438	−0.19	171
1011 — 2000	31.9168	−0.26	154
1100 — 2400	38.4	-	128
1101 — 4800	76.8	-	64
1110 — 9600	153.6	-	32
1111 — 19200	307.2	-	16

Timing
The EPCI contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full duplex operation. See table 1.

Receiver
The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU.

Transmitter
The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin.

Modem Control
The modem control section provides interfacing for three input signals and three output signals used for "handshaking" and status indication between the CPU and a modem.

SYN/DLE Control
This section contains control circuitry and three 8-bit registers storing the SYN1, SYN2, and DLE characters provided by the CPU. These registers are used in the synchronous mode of operation to provide the characters required for synchronization, idle fill and data transparency.

Table 1 BAUD RATE GENERATOR CHARACTERISTICS (Cont'd)

SCN2661B (BRCLK = 4.9152MHz)

MR23-20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	45.5	0.7279kHz	0.005	6752
0001	50	0.8	-	6144
0010	75	1.2	-	4096
0011	110	1.7598	−0.01	2793
0100	134.5	2.152	-	2284
0101	150	2.4	-	2048
0110	300	4.8	-	1024
0111	600	9.6	-	512
1000	1200	19.2	-	256
1001	1800	28.7438	−0.19	171
1010	2000	31.9168	−0.26	154
1011	2400	38.4	-	128
1100	4800	76.8	-	64
1101	9600	153.6	-	32
1110	19200	307.2	-	16
1111	38400	614.4	-	8

SCN2661C (BRCLK = 5.0688MHz)

MR23-20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	-	6336
0001	75	1.2	-	4224
0010	110	1.76	-	2880
0011	134.5	2.1523	0.016	2355
0100	150	2.4	-	2112
0101	300	4.8	-	1056
0110	600	9.6	-	528
0111	1200	19.2	-	264
1000	1800	28.8	-	176
1001	2000	32.081	0.253	158
1010	2400	38.4	-	132
1011	3600	57.6	-	88
1100	4800	76.8	-	66
1101	7200	115.2	-	44
1110	9600	153.6	-	33
1111	19200	316.8	3.125	16

NOTE
16X clock is used in asynchronous mode. In synchronous mode, clock multiplier is 1X and BRG can be used only for TxC.

Table 2 CPU-RELATED SIGNALS

PIN NAME	24-PIN	28-PIN	INPUT/OUTPUT	FUNCTION
RESET	X	X	I	A high on this input performs a master reset on the 2661. This signal asynchronously terminates any device activity and clears the mode, command and status registers. The device assumes the idle state and remains there until initialized with the appropriate control words.
A_1-A_0	X	X	I	Address lines used to select internal EPCI registers.
$\overline{R}/W$	X	X	I	Read command when low, write command when high.
$\overline{CE}$	X	X	I	Chip enable command. When low, indicates that control and data lines to the EPCI are valid and that the operation specified by the $\overline{R}/W$, A_1 and A_0 inputs should be performed. When high, places the D_0-D_7 lines in the three-state condition.
D_7-D_0	X	X	I/O	8-bit, three-state data bus used to transfer commands, data and status between EPCI and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{TxRDY}$		X	O	This output is the complement of status register bit SR0. When low, it indicates that the transmit data holding register (THR) is ready to accept a data character from the CPU. It goes high when the data character is loaded. This output is valid only when the transmitter is enabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{RxRDY}$		X	O	This output is the complement of status register bit SR1. When low, it indicates that the receive data holding register (RHR) has a character ready for input to the CPU. It goes high when the RHR is read by the CPU, and also when the receiver is disabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{TxEMT}$/ $\overline{DSCHG}$		X	O	This output is the complement of status register bit SR2. When low, it indicates that the transmitter has completed serialization of the last character loaded by the CPU, or that a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs has occurred. This output goes high when the status register is read by the CPU, if the TxEMT condition does not exist. Otherwise, the THR must be loaded by the CPU for this line to go high. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{INTR}$	X		O	This is an active low output which is the wire-OR of the $\overline{TxRDY}$, $\overline{RxRDY}$, and $\overline{TxEMT}$/$\overline{DSCHG}$ outputs on the 28-pin version. See above.

OPERATION

The functional operation of the 2661 is programmed by a set of control words supplied by the CPU. These control words specify items such as synchronous or asynchronous mode, baud rate, number of bits per character, etc. The programming procedure is described in the EPCI programming section of the data sheet.

After programming, the EPCI is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The 2661 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. In the asynchronous mode, the receiver looks for a high to low (mark to space) transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and one stop bit have been assembled. The data are then transferred to the receive data holding register, the RxRDY bit in the status register is set, and the $\overline{RxRDY}$ output is asserted. If the character length is less than 8 bits, the high order unused bits in the holding register are set to zero. The parity error, framing error, and overrun error status bits are strobed into the status register on the positive going edge of $\overline{RxC}$ corresponding to the received character boundary. If the stop bit is present, the receiver will immediately begin its search for the next start bit. If the stop bit is absent (framing error), the receiver will interpret a space as a start bit if it persists into the next bit time interval. If a break condition is detected (RxD is low for the entire character as well as the stop bit), only one character consisting of all zeros (with the FE status bit SR5 set) will be transferred to the holding register. The RxD input must return to a high condition before a search for the next start bit begins.

Pin 25 can be programmed to be a break detect output by appropriate setting of MR27-MR24. If so, a detected break will cause that pin to go high. When RxD returns to mark for one RxC time, pin 25 will go low. Refer to the break detection timing diagram.

Table 3 DEVICE-RELATED SIGNALS

PIN NAME	24-PIN	28-PIN	INPUT/OUTPUT	FUNCTION
BRCLK	X	X	I	Clock input to the internal baud rate generator (see table 1). Not required if external receiver and transmitter clocks are used.
$\overline{RxC}$/BKDET	X	X	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. Data are sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin can be a 1X/16X clock or a break detect output pin.
$\overline{TxC}$/XSYNC	X	X	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, this pin can be a 1X/16X clock output or an external jam synchronization input.
RxD	X	X	I	Serial data input to the receiver. "Mark" is high, "space" is low.
TxD	X	X	O	Serial data output from the transmitter. "Mark" is high, "space" is low. Held in mark condition when the transmitter is disabled.
$\overline{DSR}$		X	I	General purpose input which can be used for data set ready or ring indicator condition. Its complement appears as status register bit SR7. Causes a low output on $\overline{TxEMT}$/$\overline{DSCHG}$ when its state changes if CR2 or CR0 = 1.
$\overline{DCD}$	X	X	I	Data carrier detect input. Must be low in order for the receiver to operate. Its complement appears as status register bit SR6. Causes a low output on $\overline{TxEMT}$/$\overline{DSCHG}$ when its state changes if CR2 or CR0 = 1. If $\overline{DCD}$ goes high while receiving, the RxC is internally inhibited.
$\overline{CTS}$	X	X	I	Clear to send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the transmit shift register will be transmitted before termination.
$\overline{DTR}$		X	O	General purpose output which is the complement of command register bit CR1. Normally used to indicate data terminal ready.
$\overline{RTS}$	X	X	O	General purpose output which is the complement of command register bit CR5. Normally used to indicate request to send. If the transmit shift register is not empty when CR5 is reset (1 to 0), then $\overline{RTS}$ will go high one TxC time after the last serial bit is transmitted.

When the EPCI is initialized into the synchronous mode, the receiver first enters the hunt mode on a 0 to 1 transition of RxEN(CR2). In this mode, as data are shifted into the receiver shift register a bit at a time, the contents of the register are compared to the contents of the SYN1 register. If the two are not equal, the next bit is shifted in and the comparison is repeated. When the two registers match, the hunt mode is terminated and character assembly mode begins. If single SYN operation is programmed, the SYN DETECT status bit is set. If double SYN operation is programmed, the first character assembled after SYN1 must be SYN2 in order for the SYN DETECT bit to be set. Otherwise, the EPCI returns to the hunt mode. (Note that the sequence SYN1-SYN1-SYN2 will not achieve synchronization.) When synchronization has been achieved, the EPCI continues to assemble characters and transfer them to the holding register, setting the RxRDY status bit and asserting the $\overline{RxRDY}$ output each time a character is transferred. The PE and OE status bits are set as appropriate. Further receipt of the appropriate SYN sequence sets the SYN DETECT status bit. If the SYN stripping mode is commanded, SYN characters are not transferred to the holding register. Note that the SYN characters used to establish initial synchronization are not transferred to the holding register in any case.

External jam synchronization can be achieved via pin 9 by appropriate setting of MR27-MR24. When pin 9 is an XSYNC input, the internal SYN1, SYN1-SYN2, and DLE-SYN1 detection is disabled. Each positive going signal on XSYNC will cause the receiver to establish synchronization on the rising edge of the next RxC pulse. Character assembly will start with the RxD input at this edge. XSYNC may be lowered on the next rising edge of RxC. This external synchronization will cause the SYN DETECT status bit to be set until the status register is read. Refer to XSYNC timing diagram.

Transmitter

The EPCI is conditioned to transmit data when the $\overline{CTS}$ input is low and the TxEN command register bit is set. The 2661 indicates to the CPU that it can accept a character for transmission by setting the TxRDY status bit and asserting the $\overline{TxRDY}$ output. When the CPU writes a character into the transmit data holding register, these conditions are negated. Data are transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

In the asynchronous mode, the transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the transmit holding register, the TxD output remains in the marking (high) condition and the $\overline{TxEMT}/\overline{DSCHG}$ output and its corresponding status bit are asserted. Transmission resumes when the CPU loads a new character into the holding register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the send break command bit (CR3) high.

In the synchronous mode, when the 2661 is initially conditioned to transmit, the TxD output remains high and the TxRDY condition is asserted until the first character to be transmitted (usually a SYN character) is loaded by the CPU. Subsequent to this, a continuous stream of characters is transmitted. No extra bits (other than parity, if commanded) are generated by the EPCI unless the CPU fails to send a new character to the EPCI by the time the transmitter has completed sending the previous character. Since synchronous communication does not allow gaps between characters, the EPCI asserts TxEMT and automatically "fills" the gap by transmitting SYN1s, SYN1-SYN2 doublets, or DLE-SYN1 doublets, depending on the state of MR16 and MR17. Normal transmission of the message resumes when a new character is available in the transmit data holding register. If the SEND DLE bit in the command register is true, the DLE character is automatically transmitted prior to transmission of the message character in the THR.

EPCI PROGRAMMING

Prior to initiating data communications, the 2661 operational mode must be programmed by performing write operations to the mode and command registers. In addition, if synchronous operation is programmed, the appropriate SYN/DLE registers must be loaded. The EPCI can be reconfigured at any time during program execution. A flowchart of the intialization process appears in figure 1.

The internal registers of the EPCI are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in table 4.

The SYN1, SYN2, and DLE registers are accessed by performing write operations with the conditions $A_1 = 0$, $A_0 = 1$, and

Table 4 2661 REGISTER ADDRESSING

$\overline{CE}$	A_1	A_0	$\overline{R}/W$	FUNCTION
1	X	X	X	Three-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Write SYN1/SYN2/DLE registers
0	1	0	0	Read mode registers ½
0	1	0	1	Write mode registers ½
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE
See AC characteristics section for timing requirements.

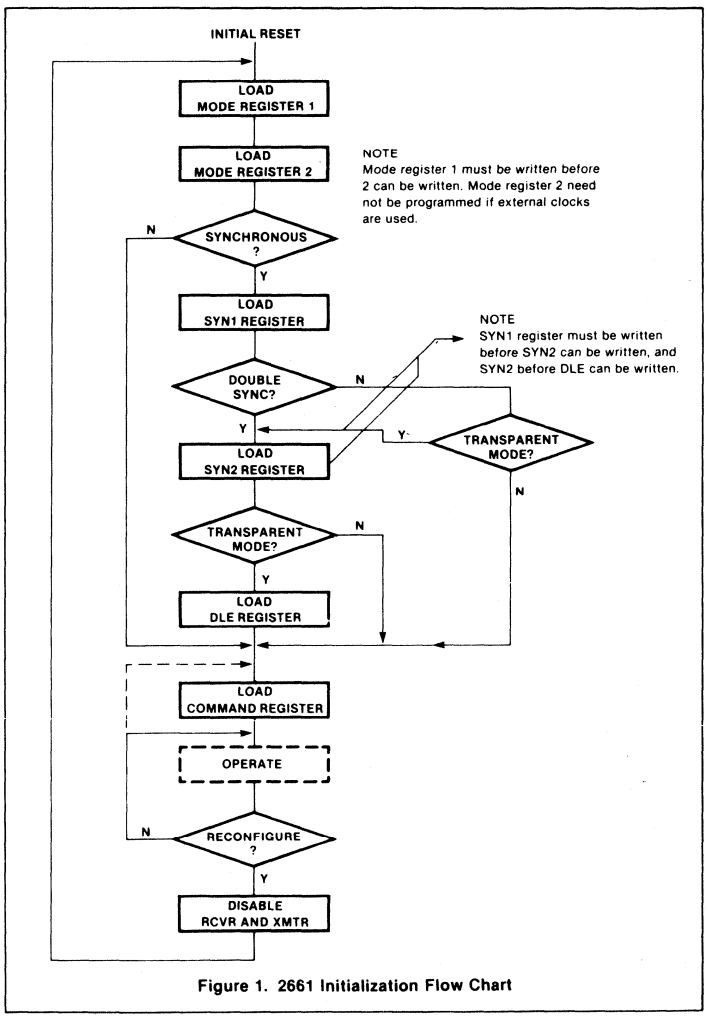


Figure 1. 2661 Initialization Flow Chart

$\overline{R}/W$ = 1. The first operation loads the SYN1 register. The next loads the SYN2 register, and the third loads the DLE register. Reading or loading the mode registers is done in a similar manner. The first write (or read) operation addresses mode register 1, and a subsequent operation addresses mode register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointers are reset to SYN1 register and mode register 1 by a RESET input or by performing a read command register operation, but are unaffected by any other read or write operation.

The 2661 register formats are summarized in tables 5, 6, 7 and 8. Mode registers 1 and 2 define the general operational characteristics of the EPCI, while the command register controls the operation within this basic framework. The EPCI indicates its status in the status register. These registers are cleared when a RESET input is applied.

Mode Register 1 (MR1)

Table 5 illustrates Mode Register 1. Bits MR11 and MR10 select the communication format and baud rate multiplier. 00 specifies synchronous mode and 1X multiplier. 1X, 16X, and 64X multipliers are programmable for asynchronous format. However, the multiplier in asynchronous format applies only if the external clock input option is selected by MR24 or MR25.

MR13 and MR12 select a character length of 5, 6, 7 or 8 bits. The character length does not include the parity bit, if programmed, and does not include the start and stop bits in asynchronous mode.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

In asynchronous mode, MR17 and MR16 select character framing of 1, 1.5, or 2 stop bits. (If 1X baud rate is programmed, 1.5 stop bits defaults to 1 stop bits on transmit.) In synchronous mode, MR17 controls the number of SYN characters used to establish synchronization and for character fill when the transmitter is idle. SYN1 alone is used if MR17 = 1, and SYN1-SYN2 is used when MR17 = 0. If the transparent mode is specified by MR16, DLE-SYN1 is used for character fill and SYN detect, but the normal synchronization sequence is used to establish character sync. When transmitting, a DLE character in the transmit holding register will cause a second DLE character to be transmitted. This DLE stuffing eliminates the software DLE compare and stuff on each transparent mode data character. If the send DLE command (CR3) is active when a DLE is loaded into THR, only one addjtional DLE will be transmitted. Also, DLE stripping and DLE detect (with MR14 = 0) are enabled.

The bits in the mode register affecting character assembly and disassembly (MR12-MR16) can be changed dynamically (during active receive/transmit operation). The character mode register affects both the transmitter and receiver; therefore in synchronous mode, changes should be made only in half duplex mode (RxEN = 1 or TxEN = 1, but not both simultaneously = 1). In asynchronous mode, character changes should be made when RxEN and TxEN=0 or when TxEN = 1 and the transmitter is marking in half duplex mode (RxEN = 0).

To effect assembly/disassembly of the next received/transmitted character, MR12-15 must be changed within n bit times of the active going state of $\overline{RxRDY}/\overline{TxRDY}$. Transparent and non-transparent mode changes (MR16) must occur within n-1 bit times of the character to be affected when the receiver or transmitter is active. (n = smaller of the new and old character lengths.)

Mode Register 2 (MR2)

Table 6 illustrates mode register 2. MR23, MR22, MR21 and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable for each EPCI version (−1, −2, −3). Version 1 and 2 specify a 4.9152 MHz TTL input at BRCLK (pin 20); version 3 specifies a 5.0688 MHz input which is identical to the Signetics 2651. MR23-20 are don't cares if external clocks are selected (MR25-MR24 = 0). The individual rates are given in table 1.

MR24-MR27 select the receive and transmit clock source (either the BRG or an external input) and the function at pins 9 and 25. Refer to table 6.

Command Register (CR)

Table 7 illustrates the command register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. A 0 to 1 transition of CR2 forces start bit search (async mode) or hunt mode (sync mode) on the second $\overline{RxC}$ rising edge. Disabling the receiver causes $\overline{RxRDY}$ to go high (inactive). If the transmitter is disabled, it will complete the transmission of the character in the transmit shift register (if any) prior to terminating operation. The TxD output will then remain in the marking state

Table 5 MODE REGISTER 1 (MR 1)

MR17 MR16		MR15	MR14	MR13 MR12	MR11 MR10
Sync/Async		Parity Type	Parity Control	Character Length	Mode and Baud Rate Factor
Async: Stop Bit Length 00 = Invalid 01 = 1 stop bit 10 = 1½ stop bits 11 = 2 stop bits		0 = Odd 1 = Even	0 = Disabled 1 = Enabled	00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits	00 = Synchronous 1X rate 01 = Asynchronous 1X rate 10 = Asynchronous 16X rate 11 = Asynchronous 64X rate
Sync: Number of SYN char 0 = Double SYN 1 = Single SYN	**Sync: Transparency Control** 0 = Normal 1 = Transparent				

NOTE
Baud rate factor in asynchronous applies only if external clock is selected. Factor is 16X if internal clock is selected. Mode must be selected (MR11, MR10) in any case.

Table 6 MODE REGISTER 2 (MR2)

MR27–MR24											MR23–MR20
	TxC	RxC	Pin 9	Pin 25		TxC	RxC	Pin 9	Pin 25	Mode	Baud Rate Selection
0000	E	E	TxC	RxC	1000	E	E	XSYNC[1]	RxC/TxC	sync	
0001	E	I	TxC	1X	1001	E	I	TxC	BKDET	async	
0010	I	E	1X	RxC	1010	I	E	XSYNC[1]	RxC	sync	
0011	I	I	1X	1X	1011	I	I	1X	BKDET	async	See baud rates in table 1
0100	E	E	TxC	RxC	1100	E	E	XSYNC[1]	RxC/TxC	sync	
0101	E	I	TxC	16X	1101	E	I	TxC	BKDET	async	
0110	I	E	16X	RxC	1110	I	E	XSYNC[1]	RxC	sync	
0111	I	I	16X	16X	1111	I	I	16X	BKDET	async	

NOTES:
1. When pin 9 is programmed as XSYNC input, SYN1, SYN1-SYN2, and DLE-SYN1 detection is disabled.

E = External clock
I = Internal clock (BRG)
1X and 16X are clock outputs

Table 7 COMMAND REGISTER (CR)

CR7 CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode	**Request To Send**	**Reset Error**	**Sync/Async**	**Receive Control (RxEN)**	**Data Terminal Ready**	**Transmit Control (TxEN)**
00 = Normal operation 01 = Async: Automatic echo mode Sync: SYN and/or DLE stripping mode 10 = Local loop back 11 = Remote loop back	0 = Force $\overline{RTS}$ output high one clock time after TxSR serialization 1 = Force $\overline{RTS}$ output low	0 = Normal 1 = Reset error flags in status register (FE, OE, PE/DLE detect.)	**Async:** **Force break** 0 = Normal 1 = Force break	0 = Disable 1 = Enable	0 = Force $\overline{DTR}$ output high 1 = Force $\overline{DTR}$ output low (Not applicable in 24-pin version.)	0 = Disable 1 = Enable
			Sync: **Send DLE** 0 = Normal 1 = Send DLE			

Table 8 STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Data Set Ready	**Data Carrier Detect**	**FE/SYN Detect**	**Overrun**	**PE DLE Detect**	**TxEMT DSCHG**	**RxRDY**	**TxRDY**
0 = $\overline{DSR}$ input is high 1 = $\overline{DSR}$ input is low (Should be ignored in 24-pin version.)	0 = $\overline{DCD}$ input is high 1 = $\overline{DCD}$ input is low	**Async:** 0 = Normal 1 = Framing Error	0 = Normal 1 = Overrun Error	**Async:** 0 = Normal 1 = Parity error	0 = Normal 1 = Change in $\overline{DSR}$ (28-pin version only), or $\overline{DCD}$, or transmit shift register is empty	0 = Receive holding register empty 1 = Receive holding register has data	0 = Transmit holding register busy 1 = Transmit holding register empty
		Sync: 0 = Normal 1 = SYN detected		**Sync:** 0 = Normal 1 = Parity error or DLE received			

(high) while $\overline{TxRDY}$ and $\overline{TxEMT}$ will go high (inactive). If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be neglected. A 0 to 1 transition of CR2 will initiate start bit search (async) or hunt mode (sync).

Bits CR1 (28-pin only) (DTR) and CR5 (RTS) control the $\overline{DTR}$ and $\overline{RTS}$ outputs. Data at the outputs are the logical complement of the register data.

In asynchronous mode, setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The TxD line will go high for at least one bit time before beginning transmission of the next character in the transmit data holding register. In synchronous mode, setting CR3 causes the transmission of the DLE register contents prior to sending the character in the transmit data holding register. Since this is a one time command, CR3 does not have to be reset by software. CR3 should be set when entering and exiting transparent mode and for all DLE—non-DLE character sequences.

Setting CR4 causes the error flags in the status register (SR3, SR4, and SR5) to be cleared. This is a one time command. There is no internal latch for this bit.

Table 9 SCN2661 EPCI vs SCN2651 PCI

FEATURE	EPCI	PCI
1. MR2 Bit 6, 7	Control pin 9, 25	Not used
2. DLE detect-SR3	SR3 = 0 for DLE-DLE, DLE-SYNC1	SR3 = 1 for DLE-DLE, DLE-SYNC1
3. Reset of SR3, DLE detect	Second character after DLE, or receiver disable, or CR4 = 1	Receiver disable, or CR4 = 1
4. Send DLE-CR3	One time command	Reset via CR3 on next $\overline{\text{TxRDY}}$
5. DLE stuffing in transparent mode	Automatic DLE stuffing when DLE is loaded except if CR3 = 1	None
6. SYNC1 stripping in double sync non-transparent mode	All SYNC1	First SYNC1 of pair
7. Baud rate versions	Three	One
8. Terminate ASYNC transmission (drop RTS)	Reset CR5 in response to $\overline{\text{TxRDY}}$ changing from 1 to 0	Reset CR0 when $\overline{\text{TxEMT}}$ goes from 1 to 0. Then reset CR5 when $\overline{\text{TxEMT}}$ goes from 0 to 1
9. Break detect	Pin 25[1]	FE and null character
10. Stop bit searched	One	Two
11. External jam sync	Pin 9[2]	No
12. Data bus timing	Improved over 2651	—
13. Data bus drivers	Sink 2.2mA Source 400μA	Sink 1.6mA Source 100μA

NOTES
1. Internal BRG used for RxC.
2. Internal BRG used for TxC.

When CR5 (RTS) is set, the $\overline{\text{RTS}}$ pin is forced low. A 1 to 0 transition of CR5 will cause $\overline{\text{RTS}}$ to go high (inactive) one TxC time after the last serial bit has been transmitted (if the transmit shift register was not empty).

The EPCI can operate in one of four sub-modes within each major mode (synchronous or asynchronous). The operational sub-mode is determined by CR7 and CR6. CR7-CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the mode and status register instructions.

In asynchronous mode, CR7-CR6 = 01 places the EPCI in the automatic echo mode. Clocked, regenerated received data are automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU to receiver communications continues normally, but the CPU to transmitter link is disabled. Only the first character of a break condition is echoed. The TxD output will go high until the next valid start is detected. The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. $\overline{\text{TxRDY}}$ output = 1.
4. The $\overline{\text{TxEMT}}$/$\overline{\text{DSCHG}}$ pin will reflect only the data set change condition.
5. The TxEN command (CR0) is ignored.

In synchronous mode, CR7-CR6 = 01 places the EPCI in the automatic SYN/DLE stripping mode. The exact action taken depends on the setting of bits MR17 and MR16:

1. In the non-transparent, single SYN mode (MR17-MR16 = 10), characters in the data stream matching SYN1 are not transferred to the receive data holding register (RHR).
2. In the non-transparent, double SYN mode (MR17-MR16 = 00), characters in the data stream matching SYN1, or SYN2 if immediately preceded by SYN1, are not transferred to the RHR.
3. In transparent mode (MR16 = 1), characters in the data stream matching DLE, or SYN1 if immediately preceded by DLE, are not transferred to the RHR. However, only the first DLE of a DLE-DLE pair is stripped.

Note that automatic stripping mode does not affect the setting of the DLE detect and SYN detect status bits (SR3 and SR5).

Two diagnostic sub-modes can also be configured. In local loop back mode (CR7-CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{\text{DTR}}$ is connected to $\overline{\text{DCD}}$ and $\overline{\text{RTS}}$ is connected to $\overline{\text{CTS}}$.
3. The receiver is clocked by the transmit clock.
4. The $\overline{\text{DTR}}$, $\overline{\text{RTS}}$ and $\overline{\text{TxD}}$ outputs are held high.
5. The $\overline{\text{CTS}}$, $\overline{\text{DCD}}$, $\overline{\text{DSR}}$ and RxD inputs are ignored.

Additional requirements to operate in the local-loop back mode are that CR0 (TxEN), CR1 (DTR), and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the EPCI.

The second diagnostic mode is the remote loop back mode (CR7-CR6 = 11). In this mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data are sent to the local CPU, but the error status conditions (PE, FE) are set.
4. The $\overline{\text{RxRDY}}$, $\overline{\text{TxRDY}}$, and $\overline{\text{TxEMT}}$/$\overline{\text{DSCHG}}$ outputs are held high.
5. CR1 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the status register (as shown in table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the transmitter ready (TxRDY) status bit. It, and its corresponding output, are valid only when the transmitter is enabled. If equal to 0, it indicates that the transmit data holding register has been loaded by the CPU and the data has not been transferred to the transmit shift register. If set equal to 1, it indicates that the holding register is ready to accept data from the CPU. This bit is initially set when the transmitter is enabled by CR0, unless a character has previously been loaded into the holding register. It is not set when the automatic echo or remote loopback modes are programmed. When this bit is set, the $\overline{\text{TxRDY}}$ output pin is low. In

the automatic echo and remote loop back modes, the output is held high.

SR1, the receiver ready (RxRDY) status bit, indicates the condition of the receive data holding register. If set, it indicates that a character has been loaded into the holding register from the receive shift register and is ready to be read by the CPU. If equal to zero, there is no new character in the holding register. This bit is cleared when the CPU reads the receive data holding register or when the receiver is disabled by CR2. When set, the $\overline{\text{RxRDY}}$ output is low.

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{\text{DSR}}$ (28-pin only) or $\overline{\text{DCD}}$ inputs (when CR2 or CR0 = 1) or that the transmit shift register has completed transmission of a character and no new character has been loaded into the transmit data holding register. Note that in synchronous mode this bit will be set even though the appropriate "fill" character is transmitted. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the transmit data holding register. The DSCHG condition is enabled when TxEN = 1 or RxEN = 1. It is cleared when the status register is read by the CPU. If the status register is read twice and SR2 = 1 while SR6 and SR7 remain unchanged, then a TxEMT condition exists. When SR2 is set, the $\overline{\text{TxEMT}}/\overline{\text{DSCHG}}$ output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. In synchronous transparent mode (MR16 = 1), with parity disabled, it indicates that a character matching DLE register was received and the present character is neither SYN1 nor DLE. This bit is cleared when the next character following the above sequence is loaded into RHR, when the receiver is disabled, or by a reset error command, CR4.

The overrun error status bit, SR4, indicates that the previous character loaded into the receive holding register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled or by the reset error command, CR4.

In asynchronous mode, bit SR5 signifies that the received character was not framed by a stop bit, i.e., only the first stop bit is checked. If RHR = 0 when SR5 = 1, a break condition is present. In synchronous non-transparent mode (MR16 = 0), it indicates receipt of the SYN1 character in single SYN mode or the SYN1-SYN2 pair in double SYN mode. In synchronous transparent mode (MR16 = 1), this bit is set upon detection of the initial synchronizing characters (SYN1 or SYN1-SYN2) and, after synchronization has been achieved, when a DLE-SYN1 pair is received. The bit is reset when the receiver is disabled, when the reset error command is given in asynchronous mode, or when the status register is read by the CPU in the synchronous mode.

SR6 and SR7 (28-pin only) reflect the conditions of the $\overline{\text{DCD}}$ and $\overline{\text{DSR}}$ inputs respectively. A low input sets its corresponding status bit, and a high input clears it.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	Note 4	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Input voltage					V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
	Output voltage					V
V_{OL}	Low	I_{OL} = 2.2mA			0.4	
V_{OH}[7]	High	I_{OH} = −400μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to 5.5 V			10	μA
	3-state output leakage current					μA
I_{LH}	Data bus high	V_O = 4.0V			10	
I_{LL}	Data bus low	V_O = 0.45V			10	
I_{CC}	Power supply current				150	mA

CAPACITANCE T_A = 25°C, V_{CC} = 0V

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Capacitance					pF
C_{IN}	Input				20	
C_{OUT}	Output	fc = 1MHz			20	
$C_{I/O}$	Input/Output	Unmeasured pins tied to ground			20	

Notes on following page

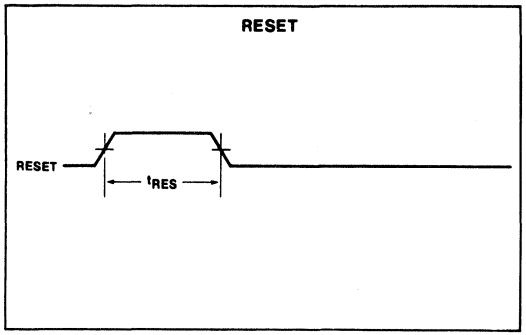
AC ELECTRICAL CHARACTERISTICS[4,5,6]

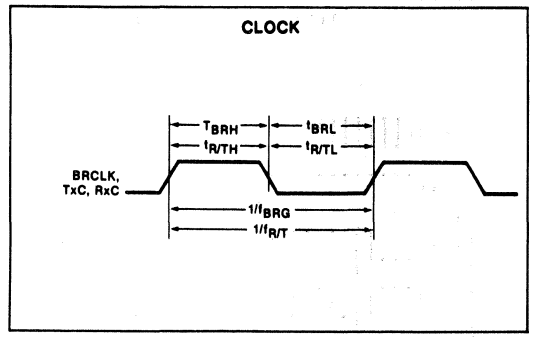
	PARAMETER	TEST CONDITIONS	Min	Typ	Max	UNIT
	Pulse width					ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		250			
	Setup and hold time					ns
t_{AS}	Address setup		10			
t_{AH}	Address hold		10			
t_{CS}	$\overline{R}/W$ control setup		10			
t_{CH}	$\overline{R}/W$ control hold		10			
t_{DS}	Data setup for write		150			
t_{DH}	Data hold for write		0			
t_{RXS}	Rx data setup		300			
t_{RXH}	Rx data hold		350			
t_{DD}	Data delay time for read	C_L = 150pF			200	ns
t_{DF}	Data bus floating time for read	C_L = 150pF			100	
t_{CED}	CE to CE delay		600			
	Input clock frequency					MHz
f_{BRG}	Baud rate generator (2661A,B)		1.0	4.9152	4.9202	
f_{BRG}	Baud rate generator (2661C)		1.0	5.0688	5.0738	
$f_{R/T}$[10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
	Clock width					ns
t_{BRH}[9]	Baud rate high (2661A,B)		75			
t_{BRH}[9]	Baud rate high (2661C)		70			
t_{BRL}[9]	Baud rate low (2661A,B)		75			
t_{BRL}[9]	Baud rate low (2661C)		70			
$t_{R/TH}$	$\overline{TxC}$ or $\overline{RxC}$ high		480			
$t_{R/TL}$[10]	$\overline{TxC}$ or $\overline{RxC}$ low		480			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	C_L = 150pF			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output [8]	C_L = 150pF		0		

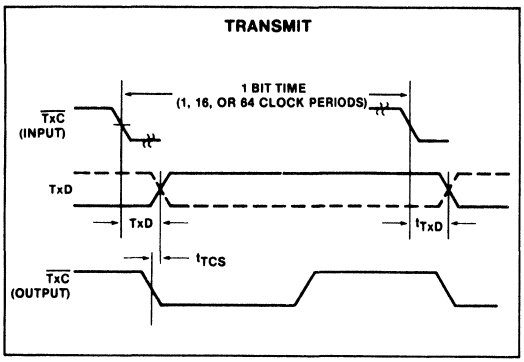
NOTES

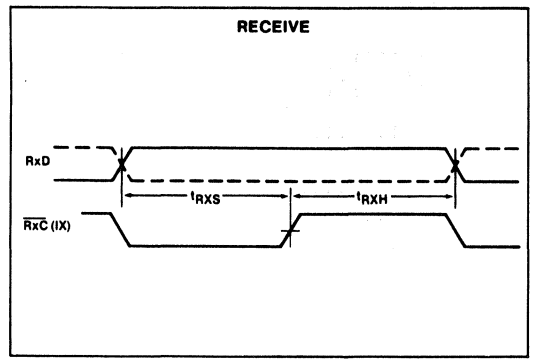
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. See ordering code table for applicable temperature range and operating supply range.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and t_{BRL}) and at 0.8V and 2.0V for outputs. Input levels swing between 0.4V and 2.4V, with a transition time of 20ns maximum.
6. Typical values are at +20°C, typical supply voltages and typical processing parameters.
7. $\overline{INTR}$, $\overline{TxRDY}$, $\overline{RxRDY}$ and $\overline{TxEMT/DSCHG}$ outputs are open drain.
8. Parameter applies when internal transmitter clock is used.
9. Under test conditions of 5.0688MHz f_{BRG} (2661C) and 4.9152MHz f_{BRG} (2661A,B), t_{BRH} and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. In asynchronous local loopback mode, using 1X clock, the following parameters apply:
    $f_{R/T}$ = 0.83MHz max.
    $t_{R/TL}$ = 700ns min.

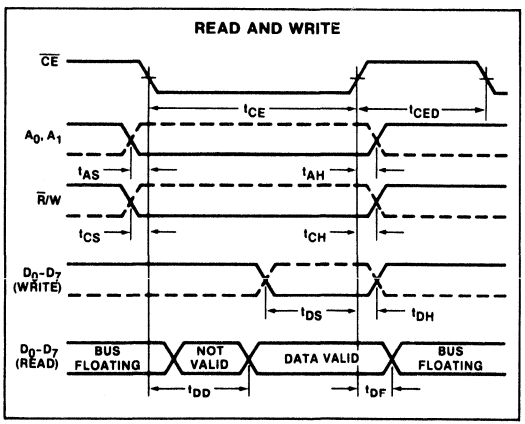
TIMING DIAGRAMS





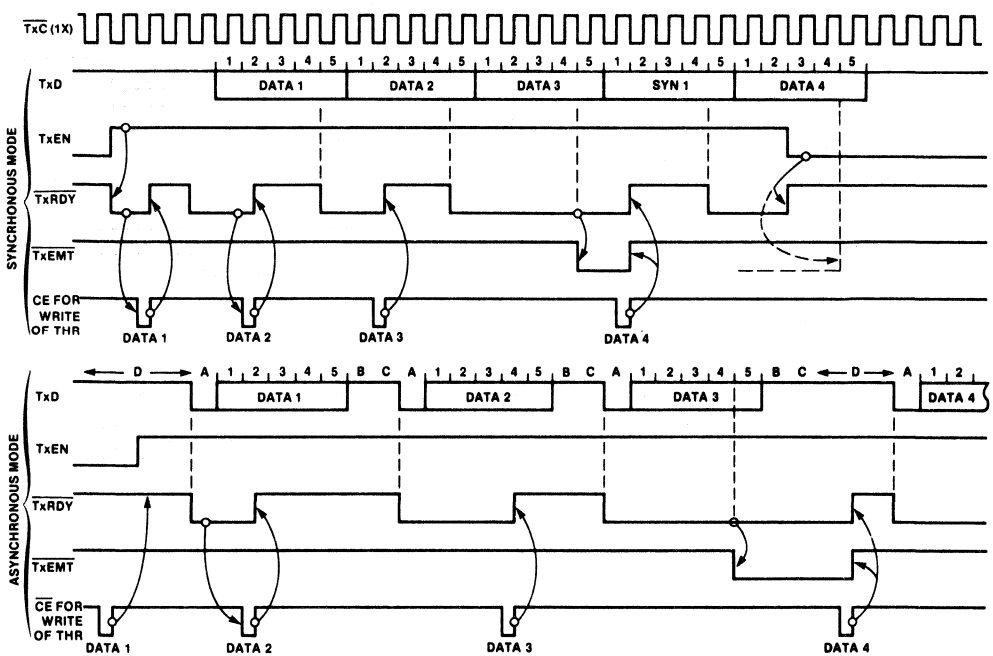






TIMING DIAGRAMS (Cont'd)

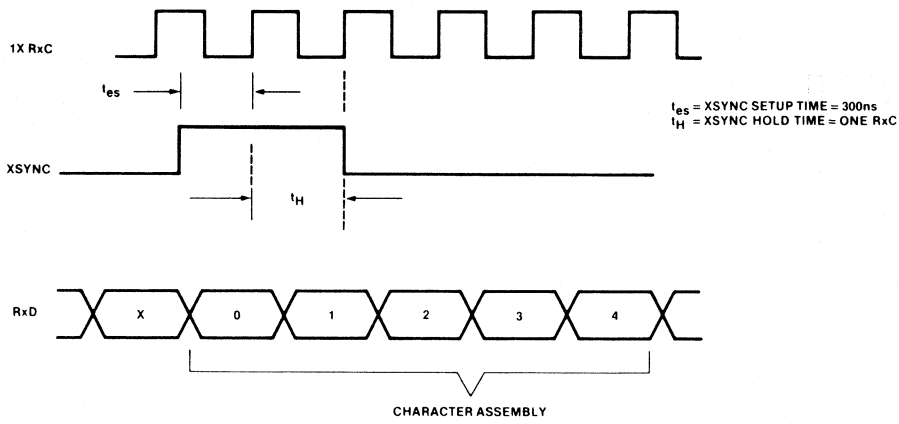
TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode])



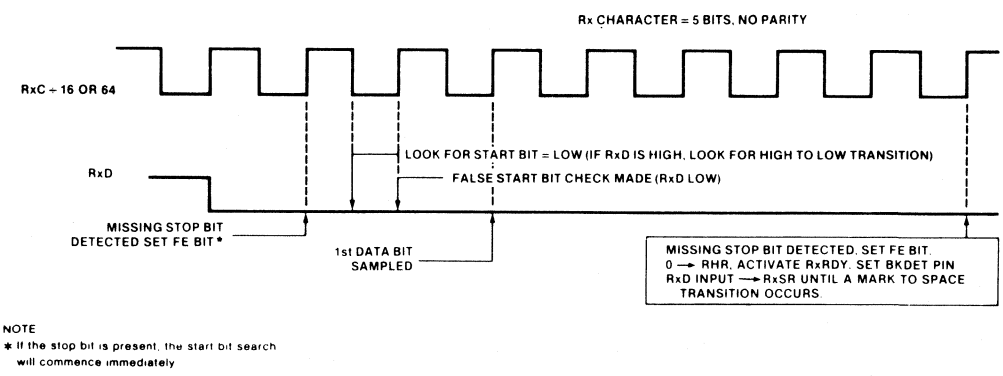
NOTES
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

TIMING DIAGRAMS (Cont'd)

EXTERNAL SYNCHRONIZATION WITH XSYNC



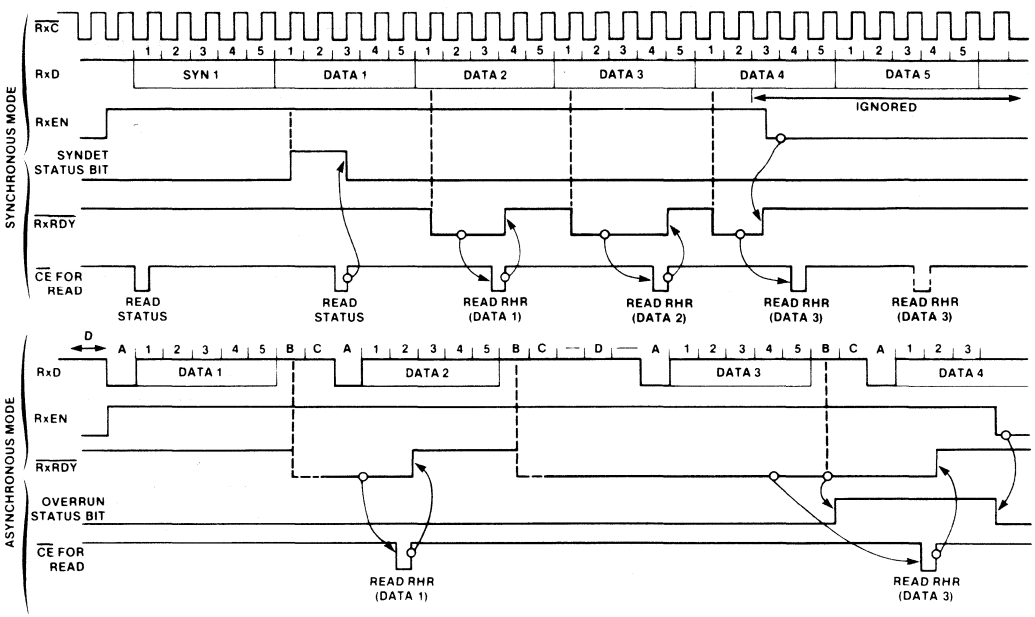
BREAK DETECTION TIMING



NOTE
* If the stop bit is present, the start bit search will commence immediately

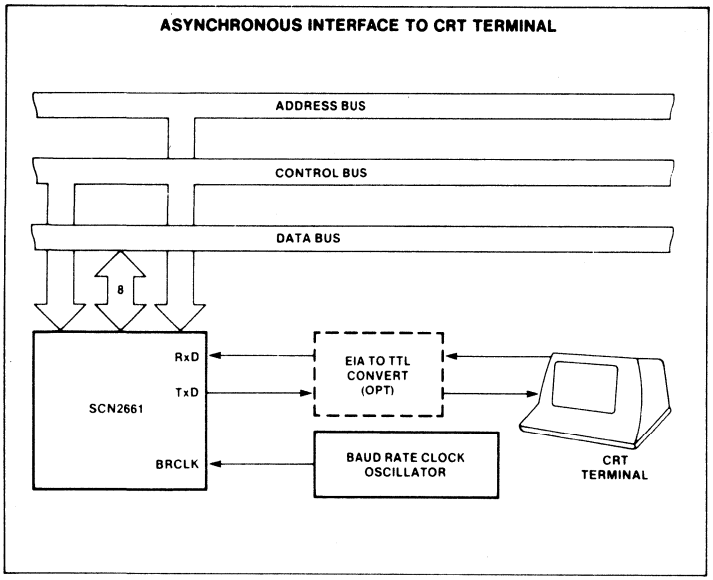
TIMING DIAGRAMS (Cont'd)

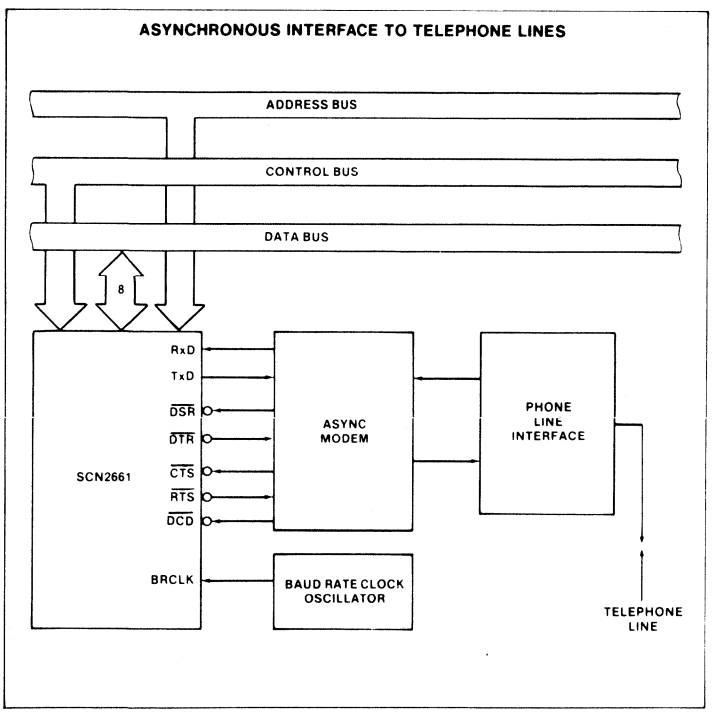
RxRDY (Shown for 5-bit characters, no parity, 2 stops bits [in asynchronous mode])



NOTES
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
Only one stop bit is detected

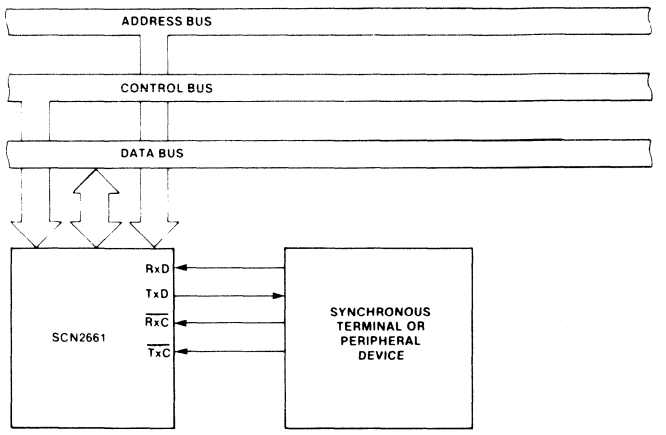
TYPICAL APPLICATIONS



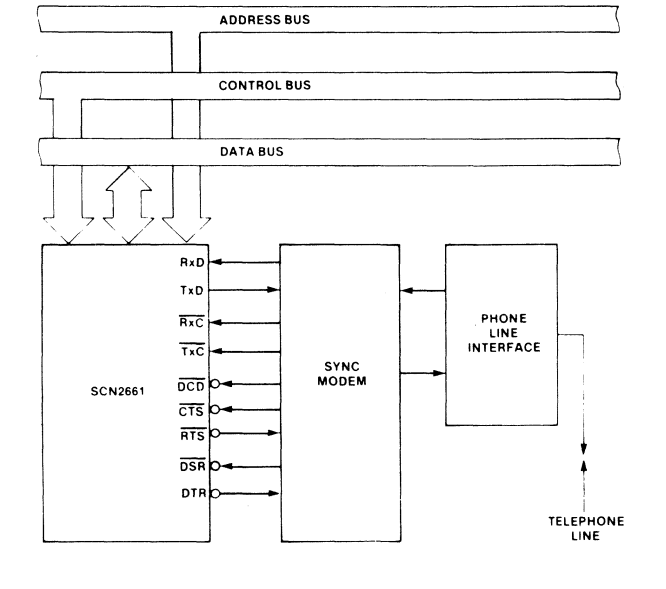


TYPICAL APPLICATIONS (Cont'd)

SYNCHRONOUS INTERFACE TO TERMINAL OR PERIPHERAL DEVICE



SYNCHRONOUS INTERFACE TO TELEPHONE LINES



DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER (DUART)

Originally published by Signetics May 1983

Preliminary

DESCRIPTION

The Signetics SCN68681 Dual Universal Asynchronous Receiver/Transmitter (DUART) is a single chip MOS-LSI communications device that provides two independent full-duplex asynchronous receiver/transmitter channels in a single package. It is compatible with other S68000 family devices, and can also interface easily with other microprocessors. The DUART can be used in polled or interrupt driven systems.

The operating mode and data format of each channel can be programmed independently. Additionally, each receiver and transmitter can select its operating speed as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the DUART particularly attractive for dual-speed channel applications such as clustered terminal systems.

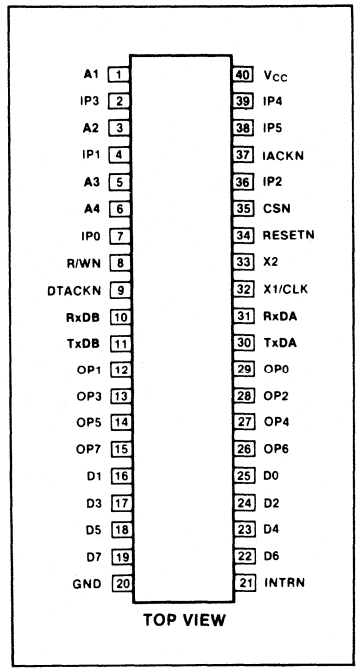
Each receiver is quadruply buffered to minimize the potential of receiver overrun or to reduce interrupt overhead in interrupt driven systems. In addition, a flow control capability is provided to disable a remote DUART transmitter when the buffer of the receiving device is full.

Also provided on the SCN68681 are a multipurpose 6-bit input port and a multipurpose 8-bit output port. These can be used as general purpose I/O ports or can be assigned specific functions (such as clock inputs or status/interrupt outputs) under program control.

FEATURES

- **S68000 bus compatible**
- **Dual full-duplex asynchronous receiver/transmiter**
- **Quadruple buffered receiver data registers**
- **Programmable data format**
  - **—5 to 8 data bits plus parity**
  - **—Odd, even, no parity or force parity**
  - **—1, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Programmable baud rate for each receiver and transmiter selectable from:**
  - **—18 fixed rates: 50 to 38.4K baud**
  - **—One user defined rate derived from programmable timer/counter**
  - **—External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **—Normal (full duplex)**
  - **—Automatic echo**
  - **—Local loopback**
  - **—Remote loopback**
- **Multi-function programmable 16-bit counter/timer**
- **Multi-function 6-bit input port**
  - **—Can serve as clock or control inputs**
  - **—Change of state detection on four inputs**
- **Multi-function 8-bit output port**
  - **—Individual bit set/reset capability**
  - **—Outputs can be programmed to be status/interrupt signals**
- **Versatile interrupt system**
  - **—Single interrupt output with eight maskable interrupting conditions**
  - **—Interrupt vector output on interrupt acknowledge**
  - **—Output port can be configured to provide a total of up to six separate wire-OR'able interrupt outputs**
- **Maximum data transfer: 1X — 1MB/sec, 16X — 125KB/sec**
- **Automatic wake-up mode for multidrop applications**
- **Start-end break interrupt/status**
- **Detects break which originates in the middle of a character**
- **On-chip crystal oscillator**
- **TTL compatible**
- **Single +5V power supply**

PIN CONFIGURATION



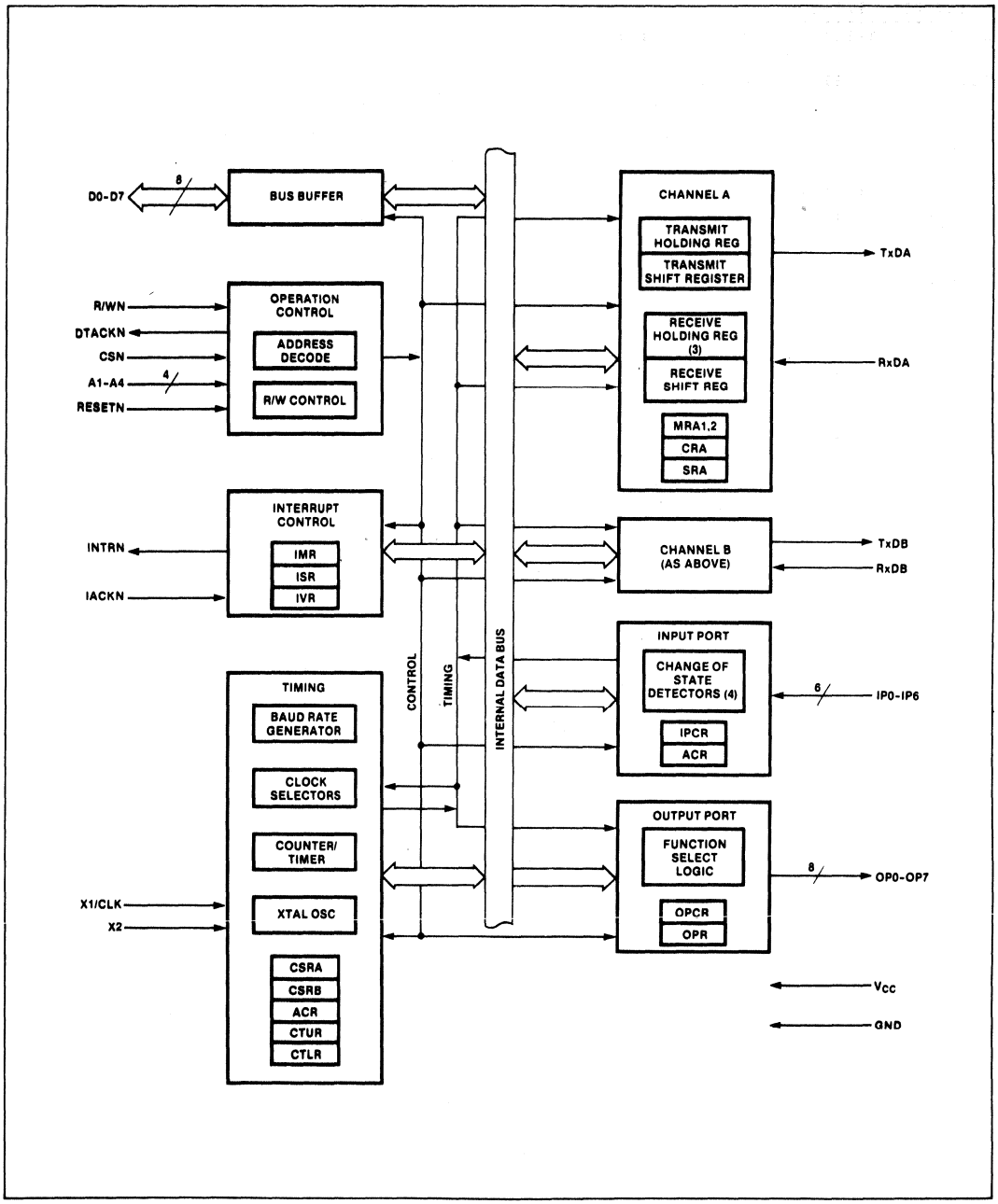
TOP VIEW

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C
Ceramic DIP	SCN68681C1I40
Plastic DIP	SCN68681C1N40

Preliminary

BLOCK DIAGRAM



Preliminary

PIN DESIGNATION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
D0-D7	25-22, 16-19	I/O	**Data Bus:** Bidirectional 3-state data bus used to transfer commands, data and status between the DUART and the CPU. D0 is the least significant bit.
CSN	35	I	**Chip Select:** Active low input signal. When low, data transfers between the CPU and the DUART are enabled on D0-D7 as controlled by the R/WN and A1-A4 inputs. When high, places the D0-D7 lines in the 3-state condition.
R/WN	8	I	**Read/Write:** A high input indicates a read cycle and a low input indicates a write cycle, when a cycle is initiated by assertion of the CSN input.
A1-A4	1, 3, 5, 6	I	**Address Inputs:** Select the DUART internal registers and ports for read/write operations.
RESETN	34	I	**Reset:** A low clears internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), initializes the IVR to hex 0F, puts OP0-OP7 in the high state, stops the counter/timer, and puts channel A and B in the inactive state, with the TxDA and TxDB outputs in the mark (high) state.
DTACKN	9	O	**Data Transfer Acknowledge:** Three-state active low output asserted in write, read, or interrupt cycles to indicate proper transfer of data between the CPU and the DUART.
INTRN	21	O	**Interrupt Request:** Active low, open drain output which signals the CPU that one or more of the eight maskable interrupting conditions are true.
IACKN	37	I	**Interrupt Acknowledge:** Active low input indicating an interrupt acknowledge cycle. In response, the DUART will place the interrupt vector on the data bus and will assert DTACKN if it has an interrupt pending.
X1/CLK	32	I	**Crystal 1:** Crystal or external clock input. A crystal or clock of the specified limits must be supplied at all times. If a crystal is used, a capacitor must be connected from this pin to ground (see figure 7).
X2	33	I	**Crystal 2:** Connection for other side of the crystal. Should be connected to ground if a crystal is not used. If a crystal is used, a capacitor must be connected from this pin to ground (see figure 7).
RxDA	31	I	**Channel A Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
RxDB	10	I	**Channel B Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
TxDA	30	O	**Channel A Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
TxDB	11	O	**Channel B Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
OP0	29	O	**Output 0.** General purpose output, or channel A request to send (RTSAN, active low). Can be deactivated automatically on receive or transmit.
OP1	12	O	**Output 1:** General purpose output, or channel B request to send (RTSBN, active low). Can be deactivated automatically on receive or transmit.
OP2	28	O	**Output 2:** General purpose output, or channel A transmitter 1X or 16X clock output, or channel A receiver 1X clock output.
OP3	13	O	**Output 3:** General purpose output, or open drain, active low counter/timer output, or channel B transmitter 1X clock output, or channel B receiver 1X clock output.
OP4	27	O	**Output 4:** General purpose output, or channel A open drain, active low, RxRDYA/FFULLA output.

Preliminary

PIN DESIGNATION (Continued)

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
OP5	14	O	**Output 5:** General purpose output, or channel B open drain, active low, RxRDYB/FFULLB output.
OP6	26	O	**Output 6:** General purpose output, or channel A open drain, active low, TxRDYA output.
OP7	15	O	**Output 7:** General purpose output, or channel B open drain, active low, TxRDYB output.
IP0	7	I	**Input 0:** General purpose input, or channel A clear to send active low input (CTSAN).
IP1	4	I	**Input 1:** General purpose input, or channel B clear to send active low input (CTSBN).
IP2	36	I	**Input 2:** General purpose input, or channel B receiver external clock input (RxCB), or counter/timer external clock input. When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
IP3	2	I	**Input 3:** General purpose input, or channel A transmitter external clock input (TxCA). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP4	39	I	**Input 4:** General purpose input, or channel A receiver external clock input (RxCA). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
IP5	38	I	**Input 5:** General purpose input, or channel B transmitter external clock input (TxCB). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
V_{CC}	40	I	**Power Supply:** +5V supply input.
GND	20	I	**Ground**

BLOCK DIAGRAM

The SCN68681 DUART consists of the following eight major sections: data bus buffer, operation control, interrupt control, timing, communications channels A and B, input port and output port. Refer to the block diagram.

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the DUART.

Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The DTACKN output is asserted during write and read cycles to indicate to the CPU that data has been latched on a write cycle, or that valid data is present on the bus on a read cycle.

Interrupt Control

A single active low interrupt output (INTRN) is provided which is activated upon the occurrence of any of eight internal events. Associated with the interrupt system are the interrupt mask register (IMR), the interrupt status register (ISR), the auxiliary control register (ACR), and the interrupt vector register (IVR). The IMR may be programmed to select only certain conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions. When IACKN is asserted, and the DUART has an interrupt pending, the DUART responds by placing the contents of the IVR register on the data bus and asserting DTACKN.

Outputs OP3-OP7 can be programmed to provide discrete interrupt outputs for the transmitters, receivers, and counter/timer.

Timing Circuits

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and four clock selectors. The crystal oscillator operates directly from a 3.6864MHz crystal connected across the X1/CLK and X2 inputs. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. The clock serves as the basic timing reference for the baud rate generator (BRG), the counter/timer, and other internal circuits. A clock signal within the limits specified in the specifications section of this data sheet must always be supplied to the DUART.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4K baud. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The four clock selectors allow the independent selection, for each receiver and transmitter, of any of these baud rates or an external timing signal.

The counter/timer (C/T) can be programmed to use one of several timing sources as its input. The output of the C/T is available to the clock selectors and can also be programmed to be output at OP3. In the counter mode, the contents of the C/T can be read by the CPU and it can be stopped and started under program control. In the timer mode, the C/T acts as a programmable divider.

Preliminary

Communications Channels A and B

Each communications channel of the SCN68681 comprises a full duplex asynchronous receiver/transmitter (UART). The operating frequency for each receiver and transmitter can be selected independently from the baud rate generator, the counter timer, or from an external input.

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate start, stop, and optional parity bits and outputs a composite serial stream of data on the TxD output pin. The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition and sends an assembled character to the CPU.

Input Port

The inputs to this unlatched 6-bit port can be read by the CPU by performing a read operation at address D_{16}. A high input results in a logic 1 while a low input results in a logic 0. D7 will always be read as a logic 1 and D6 will reflect the level of IACKN. The pins of this port can also serve as auxiliary inputs to certain portions of the DUART logic.

Four change-of-state detectors are provided which are associated with inputs IP3, IP2, IP1, and IP0. A high-to-low or low-to-high transition of these inputs lasting longer than 25-50μs will set the corresponding bit in the input port change register. The bits are cleared when the register is read by the CPU. Any change of state can also be programmed to generate an interrupt to the CPU.

Output Port

The 8-bit multi-purpose output port can be used as a general purpose output port, in which case the outputs are the complements of the output port register (OPR). OPR[n] = 1 results in OP[n] = low and vice-versa. Bits of the OPR can be individually set and reset. A bit is set by performing a write operation at address E_{16} with the accompanying data specifying the bits to be set (1 = set, 0 = no change). Likewise, a bit is reset by a write at address F_{16} with the accompanying data specifying the bits to be reset (1 = reset, 0 = no change).

Outputs can be also individually assigned specific functions by appropriate programming of the channel A mode registers (MR1A, MR2A), the channel B mode registers (MR1B, MR2B), and the output port configuration register (OPCR).

OPERATION

Transmitter

The SCN68681 is conditioned to transmit data when the transmitter is enabled through the command register. The SCN68681 indicates to the CPU that it is ready to accept a character by setting the TxRDY bit in the status register. This condition can be programmed to generate an interrupt request at OP6 or OP7 and INTRN. When a character is loaded into the transmit holding register (THR), the above conditions are negated. Data is transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again which means one full character time of buffering is provided. Characters cannot be loaded into the THR while the transmitter is disabled.

The transmitter converts the parallel data from the CPU to a serial bit stream on the TxD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TxD output remains high and the TxEMT bit in the status register (SR) will be set to 1. Transmission resumes and the TxEMT bit is cleared when the CPU loads a new character into the THR. If the transmitter is disabled, it continues operating until the character currently being transmitted is completely sent out. The transmitter can be forced to send a continuous low condition by issuing a send break command.

The transmitter can be reset through a software command. If it is reset, operation ceases immediately and the transmitter must be enabled through the command register before resuming operation. If CTS operation is enabled, the CTSN input must be low in order for the character to be transmitted. if it goes high in the middle of a transmission, the character in the shift register is transmitted and TxDA then remains in the marking state until CTSN goes low. The transmitter can also control the deactivation of the RTSN output. If programmed, the RTSN output will be reset one bit time after the character in the transmit shift register and transmit holding register (if any) are completely transmitted, if the transmitter has been disabled.

Receiver

The SCN68681 is conditioned to receive data when enabled through the command register. The receiver looks for a high to low (mark to space) transition of the start bit on the RxD input pin. If a transition is detected, the state of the RxD pin is sampled each 16X clock for 7-1/2 clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RxD is sampled high, the start bit is invalid and the search for a valid start bit begins again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least sigificant bit is received first. The data is then transferred to the receive holding register (RHR) and the RxRDY bit in the SR is set to a 1. This condition can be programmed to generate an interrupt at OP4 or OP5 and INTRN. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (framing error) and RxD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one-half bit time after the stop bit was sampled).

The parity error, framing error, overrun error and received break state (if any) are strobed into the SR at the received character boundary, before the RxRDY status bit is set. If a break condition is detected (RxD is low for the entire character including the stop bit), a character consisting of all zeros will be loaded into the RHR and the received break bit in the SR is set to 1. The RxD input must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The RHR consists of a first-in-first-out (FIFO) stack with a capacity of three characters. Data is loaded from the receive shift register into the topmost empty position of the FIFO. The RxRDY bit in the status register is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three stack positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the

Preliminary

data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits (see below) are 'popped' thus emptying a FIFO position for new data.

In addition to the data word, three status bits (parity error, framing error, and received break) are also appended to each data character in the FIFO (overrun is not). Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these three bits is the logical OR of the status for all characters coming to the top of the FIFO since the last 'reset error' command was issued. In either mode reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore the status register should be read prior to reading the FIFO.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available. If an additional character is received while this state exits, the contents of the FIFO are not affected: the character previously in the shift register is lost and the overrun error status bit (SR[4]) will be set upon receipt of the start bit of the new (overruning) character.

The receiver can control the deactivation of RTS. If programmed to operate in this mode, the RTSN output will be negated when a valid start bit was received and the FIFO is full. When a FIFO position becomes available, the RTSN output will be re-asserted automatically. This feature can be used to prevent an overrun, in the receiver, by connecting the RTSN output to the CTSN input of the transmitting device.

If the receiver is disabled, the FIFO characters can be read. However, no additional characters can be received until the receiver is enabled again. If the receiver is reset, the FIFO and all of the receiver status, and the corresponding output ports and interrupt are reset. No additional characters can be received until the receiver is enabled again.

Multidrop Mode

The DUART is equipped with a wake up mode used for multidrop applications. This mode is selected by programming bits MR1A[4:3] or MR1B[4:3] to '11' for channels A and B respectively. In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, with receivers that are normally disabled, examine the received data stream and 'wake-up' the CPU (by setting RxRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, an address/data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1A[2]/MR1B[2]. MR1A[2]/MR1B[2] = 0 transmits a zero in the A/D bit position, which identifies the corresponding data bits as data, while MR1A[2]/MR1B[2] = 1 transmits a one in the A/D bit position, which identifies the corresponding data bits as an address. The CPU should program the mode register prior to loading the corresponding data bits into the THR.

In this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RxRDY status bit and loads the character into the RHR FIFO if the received A/D bit is a one (address tag), but discards the received character if the received A/D bit is a zero (data tag). If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SRA[5] or SRB[5]). Framing error, overrun error, and break detect operate normally whether or not the receiver is enabled.

PROGRAMMING

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The addressing of the registers is described in table 1.

The contents of certain control registers are initialized to zero on RESET. Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems. For example, changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. In general, the contents of the MR, the CSR, and the OPCR should only be changed while the receiver(s) and transmitter(s) are not enabled, and certain changes to the ACR should only be made while the C/T is stopped.

Mode registers 1 and 2 of each channel are accessed via independent auxiliary pointers. The pointer is set to MR1x by RESET or by issuing a 'reset pointer' command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1x switches the pointer to MR2x. The pointer then remains at MR2x, so that subsequent accesses are always to MR2x unless the pointer is reset to MR1x as described above.

Mode, command, clock select, and status registers are duplicated for each channel to provide total independent operation and control. Refer to table 2 for register bit descriptions.

Table 1 68681 REGISTER ADDRESSING

A4	A3	A2	A1	READ (R/WN = 1)	WRITE (R/WN = 0)
0	0	0	0	Mode Register A (MR1A, MR2A)	Mode Register A (MR1A, MR2A)
0	0	0	1	Status Register A (SRA)	Clock Select Reg. A (CSRA)
0	0	1	0	*Reserved*	Command Register A (CRA)
0	0	1	1	RX Holding Register A (RHRA)	TX Holding Register A (THRA)
0	1	0	0	Input Port Change Reg. (IPCR)	Aux. Control Register (ACR)
0	1	0	1	Interrupt Status Reg. (ISR)	Interrupt Mask Reg. (IMR)
0	1	1	0	Counter/Timer Upper (CTU)	C/T Upper Register (CTUR)
0	1	1	1	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
1	0	0	0	Mode Register B (MR1B, MR2B)	Mode Register B (MR1B, MR2B)
1	0	0	1	Status Register B (SRB)	Clock Select Reg. B (CSRB)
1	0	1	0	*Reserved*	Command Register B (CRB)
1	0	1	1	RX Holding Register B (RHRB)	TX Holding Register B (THRB)
1	1	0	0	Interrupt Vector Reg. (IVR)	Interrupt Vector Reg. (IVR)
1	1	0	1	Input Port	Output Port Conf. Reg. (OPCR)
1	1	1	0	Start Counter Command	Set Output Port Bits Command
1	1	1	1	Stop Counter Command	Reset Output Port Bits Command

Preliminary

Table 2 REGISTER BIT FORMATS

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RX RTS CONTROL	RX INT SELECT	ERROR MODE	PARITY MODE		PARITY TYPE	BITS PER CHAR.	
MR1A MR1B	0 = no 1 = yes	0 = RXRDY 1 = FFULL	0 = char 1 = block	00 = with parity 01 = force parity 10 = no parity 11 = multi-drop mode		0 = even 1 = odd	00 = 5 01 = 6 10 = 7 11 = 8	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	CHANNEL MODE		Tx RTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
MR2A MR2B	00 = Normal 01 = Auto echo 10 = Local loop 11 = Remote loop		0 = no 1 = yes	0 = no 1 = yes	0 = 0.563 1 = 0.625 2 = 0.688 3 = 0.750	4 = 0.813 5 = 0.875 6 = 0.938 7 = 1.000	8 = 1.563 9 = 1.625 A = 1.688 B = 1.750	C = 1.813 D = 1.875 E = 1.938 F = 2.000

*Add 0.5 to values shown for 0-7 if channel is programmed for 5 bits/char

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
CSRA CSRB	See text				See text			

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	not used— must be 0	MISCELLANEOUS COMMANDS			DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
CRA CRB		See text			0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVED BREAK	FRAMING ERROR	PARITY ERROR	OVERRUN ERROR	TxEMT	TxRDY	FFULL	RxRDY
SRA SRB	0 = no 1 = yes *	0 = no 1 = yes *	0 = no 1 = yes *	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

*These status bits are appended to the corresponding data character in the receive FIFO. A read of the status register provides these bits (7:5) from the top of the FIFO together with bits 4:0. These bits are cleared by a 'reset error status' command. In character mode they are discarded when the corresponding data character is read from the FIFO.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	OP7	OP6	OP5	OP4	OP3		OP2	
OPCR	0 = OPR[7] 1 = TxRDYB	0 = OPR[6] 1 = TxRDYA	0 = OPR[5] 1 = RxRDY/ FFULLB	0 = OPR[4] 1 = RxRDY/ FFULLA	00 = OPR[3] 01 = C/T OUTPUT 10 = TxCB (1X) 11 = RxCB (1X)		00 = OPR[2] 01 = TxCA (16X) 10 = TxCA (1X) 11 = RxCA (1X)	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			DELTA IP3 INT	DELTA IP2 INT	DELTA IP1 INT	DELTA IP0 INT
ACR	0 = set1 1 = set2	See table 4			0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	DELTA IP3	DELTA IP2	DELTA IP1	DELTA IP0	IP3	IP2	IP1	IP0
IPCR	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high

Preliminary

Table 2 REGISTER BIT FORMATS (Continued)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ISR	INPUT PORT CHANGE	DELTA BREAK B	RxRDY/ FFULLB	TxRDYB	COUNTER READY	DELTA BREAK A	RxRDY/ FFULLA	TxRDYA
	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes
IMR	IN. PORT CHANGE INT	DELTA BREAK B INT	RxRDY/ FFULLB INT	TxRDYB INT	COUNTER READY INT	DELTA BREAK A INT	RxRDY/ FFULLA INT	TxRDYA INT
	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on
CTUR	C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]
CTLR	C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]
IVR	IVR[7]	IVR[6]	IVR[5]	IVR[4]	IVR[3]	IVR[2]	IVR[1]	IVR[0]

MR1A — Channel A Mode Register 1

MR1A is accessed when the channel A MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRA. After reading or writing MR1A, the pointer will point to MR2A.

MR1A[7] — Channel A Receiver Request-to-Send Control — This bit controls the deactivation of the RTSAN output (OP0) by the receiver. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR1A[7] = 1 causes RTSAN to be negated upon receipt of a valid start bit if the channel A FIFO is full. However, OPR[0] is not reset and RTSAN will be asserted again when an empty FIFO position is available. This feature can be used for flow control to prevent overrun in the receiver by using the RTSAN output signal to control the CTSN input of the transmitting device.

MR1A[6] — Channel A Receiver Interrupt Select — This bit selects either the channel A receiver ready status (RXRDY) or the channel A FIFO full status (FFULL) to be used for CPU interrupts. It also causes the selected bit to be output on OP4 if it is programmed as an interrupt output via the OPCR.

MR1A[5] — Channel A Error Mode Select — This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break) for channel A. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these bits is the accumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last 'reset error' command for channel A was issued.

MR1A[4:3] — Channel A Parity Mode Select — If 'with parity' or 'force parity' is selected, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1A[4:3] = 11 selects channel A to operate in the special multidrop mode described in the Operation section.

MR1A[2] — Channel A Parity Type Select — This bit selects the parity type (odd or even) if the 'with parity' mode is programmed by MR1A[4:3], and the polarity of the forced parity bit if the 'force parity' mode is programmed. It has no effect if the 'no parity' mode is programmed. In the special multidrop mode it selects the polarity of the A/D bit.

MR1A[1:0] — Channel A Bits per Character Select — This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

Preliminary

MR2A — Channel A Mode Register 2

MR2A is accessed when the channel A MR pointer points to MR2, which occurs after any access to MR1A. Accesses to MR2A do not change the pointer.

MR2A[7:6] — Channel A Mode Select — Each channel of the DUART can operate in one of four modes. MR2A[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2A[7:6] = 01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The channel A TxRDY and TxEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be configured. MR2A[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TxDA output is held high.
4. The RxDA input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2A[7:6] = 11. In this mode:

1. Received data is reclocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., transmitted parity bit is as received.
5. The receiver must be enabled.
6. Character framing is not checked, and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes: if the de-selection occurs just after the receiver has sampled the stop bit (indicated in autoecho by assertion of RxRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until the entire stop bit has been retransmitted.

MR2A[5] — Channel A Transmitter Request-to-Send Control — This bit controls the deactivation of the RTSAN output (OP0) by the transmitter. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR2A[5] = 1 causes OPR[0] to be reset automatically one bit time after the characters in the channel A transmit shift register and in the THR, if any, are completely transmitted, including the programmed number of stop bits, if the transmitter is not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2A[5] = 1.
2. Enable transmitter.
3. Assert RTSAN: OPR[0] = 1.
4. Send message.
5. Disable transmitter after the last character is loaded into the channel A THR.
6. The last character will be transmitted and OPR[0] will be reset one bit time after the last stop bit, causing RTSAN to be negated.

MR2A[4] — Channel A Clear-to-Send Control — If this bit is 0, CTSAN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSAN (IP0) each time it is ready to send a character. If IP0 is asserted (low), the character is transmitted. If it is negated (high), the TxDA output remains in the marking state and the transmission is delayed until CTSAN goes low. Changes in CTSAN while a character is being transmitted do not affect the transmission of that character.

MR2A[3:0] — Channel A Stop Bit Length Select — This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, MR2A[3] = 0 selects one stop bit and MR2A[3] = 1 selects two stop bits to be transmitted.

MR1B — Channel B Mode Register 1

MR1B is accessed when the channel B MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRB. After reading or writing MR1B, the pointer will point to MR2B.

The bit definitions for this register are identical to the bit definitions for MR1A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

MR2B — Channel B Mode Register 2

MR2B is accessed when the channel B MR pointer points to MR2, which occurs after any access to MR1B. Accesses to MR2B do not change the pointer.

The bit definitions for this register are identical to the bit definitions for MR2A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

CSRA — Channel A Clock Select Register

CSRA[7:4] — Channel A Receiver Clock Select — This field selects the baud rate clock for the channel A receiver as follows:

Preliminary

CSRA[7:4]	Baud Rate Clock = 3.6864MHz ACR[7] = 0	ACR[7] = 1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1,200	1,200
0 1 1 1	1,050	2,000
1 0 0 0	2,400	2,400
1 0 0 1	4,800	4,800
1 0 1 0	7,200	1,800
1 0 1 1	9,600	9,600
1 1 0 0	38.4K	19.2K
1 1 0 1	Timer	Timer
1 1 1 0	IP4—16X	IP4—16X
1 1 1 1	IP4—1X	IP4—1X

The receiver clock is always a 16X clock except for CSRA[7:4] = 1111.

CSRA[3:0] — Channel A Transmitter Clock Select — This field selects the baud rate clock for the channel A transmitter. The field definition is as per CSRA[7:4] except as follows:

CSRA[3:0]	Baud Rate ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP3—16X	IP3—16X
1 1 1 1	IP3—1X	IP3—1X

The transmitter clock is always a 16X clock except for CSRA[3:0] = 1111.

CSRB — Channel B Clock Select Register

CSRB[7:4] — Channel B Receiver Clock Select — This field selects the baud rate clock for the channel B receiver. The field definition is as per CSRA[7:4] except as follows:

CSRB[7:4]	Baud Rate ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP2—16X	IP2—16X
1 1 1 1	IP2—1X	IP2—1X

The receiver clock is always a 16X clock except for CSRB[7:4] = 1111.

CSRB[3:0] — Channel B Transmitter Clock Select — This field selects the baud rate clock for the channel B transmitter. The field definition is as per CSRA[7:4] except as follows:

CSRB[3:0]	Baud Rate ACR[7] = 0	ACR[7] = 1
1 1 1 0	IP5—16X	IP5—16X
1 1 1 1	IP5—1X	IP5—1X

The transmitter clock is always a 16X clock except for CSRB[3:0] = 1111.

CRA — Channel A Command Register

CRA is a register used to supply commands to channel A. Multiple commands can be specified in a single write to CRA as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

CRA[6:4] — Channel A Miscellaneous Commands — The encoded value of this field may be used to specify a single command as follows:

CRA[6:4]	COMMAND
0 0 0	No command.
0 0 1	Reset MR pointer. Causes the channel A MR pointer to point to MR1.
0 1 0	Reset receiver. Resets the channel A receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.
0 1 1	Reset transmitter. Resets the channel A transmitter as if a hardware reset had been applied.
1 0 0	Reset error status. Clears the channel A Received Break, Parity Error, Framing Error, and Overrun Error bits in the status register (SRA[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared) and in block mode to clear all error status after a block of data has been received.
1 0 1	Reset channel A break change interrupt. Causes the channel A break detect change bit in the interrupt status register (ISR[2]) to be cleared to zero.
1 1 0	Start break. Forces the TXDA output low (spacing). If the transmitter is empty the start of the break condition will be delayed up to two bit times. If the transmitter is active the break begins when transmission of the character is completed. If a character is in the THR, the start of the break will be delayed until that character, or any others loaded subsequently are transmitted. The transmitter must be enabled for this command to be accepted.
1 1 1	Stop Break. The TXDA line will go high (marking) within two bit times. TXDA will remain high for one bit time before the next character, if any, is transmitted.

CRA[3] — Disable Channel A Transmitter — This command terminates transmitter operation and resets the TxRDY and TxEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CRA[2] — Enable Channel A Transmitter — Enables operation of the channel A transmitter. The TxRDY status bit will be asserted.

CRA[1] — Disable Channel A Receiver — This command terminates operation of the receiver immediately — a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special multidrop mode is programmed, the receiver operates even if it is disabled. See Operation section.

CRA[0] — Enable Channel A Receiver — Enables operation of the channel A receiver. If not in the special wakeup mode, this also forces the receiver into the search for start-bit state.

CRB — Channel B Command Register

CRB is a register used to supply commands to channel B. Multiple commands can be specified in a single write to CRB as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

The bit definitions for this register are identical to the bit definitions for CRA, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

SRA — Channel A Status Register

SRA[7] — Channel A Received Break — This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received: further entries to the FIFO are inhibited until the RxDA line returns to the marking state for at least one-half a bit time (two successive edges of the internal or external 1x clock).

Preliminary

When this bit is set, the channel A 'change in break' bit in the ISR (ISR[2]) is set. ISR[2] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry can detect breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until at least the end of the next character time in order for it to be detected.

SRA[6] — Channel A Framing Error — This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SRA[5] — Channel A Parity Error — This bit is set when the 'with parity' or 'force parity' mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special multidrop mode the parity error bit stores the received A/D bit.

SRA[4] — Channel A Overrun Error — This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a 'reset error status' command.

SRA[3] — Channel A Transmitter Empty (TxEMTA) — This bit will be set when the channel A transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. It is set after transmission of the last stop bit of a character if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU or when the transmitter is disabled.

SRA[2] — Channel A Transmitter Ready (TxRDYA) — This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TxRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled, viz., characters loaded into the THR while the transmitter is disabled will not be transmitted.

SRA[1] — Channel A FIFO Full (FFULLA) — This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, FFULL will not be reset when the CPU reads the RHR.

SRA[0] — Channel A Receiver Ready (RxRDYA) — This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR, if after this read there are no more characters still in the FIFO.

SRB — Channel B Status Register

The bit definitions for this register are identical to the bit definitions for SRA, except that all status applies to the channel B receiver and transmitter and the corresponding inputs and outputs.

OPCR — Output Port Configuration Register

OPCR[7] — OP7 Output Select — This bit programs the OP7 output to provide one of the following:

- The complement of OPR[7]
- The channel B transmitter interrupt output, which is the complement of TxRDYB. When in this mode OP7 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[6] — OP6 Output Select — This bit programs the OP6 output to provide one of the followng:

- The complement of OPR[6]
- The channel A transmitter interrupt output, which is the complement of TxRDYA. When in this mode OP6 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[5] — OP5 Output Select — This bit programs the OP5 output to provide one of the following:

- The complement of OPR[5]
- The channel B receiver interrupt output, which is the complement of ISR[5]. When in this mode OP5 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[4] — OP4 Output Select — This bit programs the OP4 output to provide one of the following:

- The complement of OPR[4]
- The channel A receiver interrupt output, which is the complement of ISR[1]. When in this mode OP4 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[3:2] — OP3 Output Select — This field programs the OP3 output to provide one of the following:

- The complement of OPR[3]
- The counter/timer output, in which case OP3 acts as an open collector output. In the timer mode, this output is a square wave at the programmed frequency. In the counter mode, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state when the counter is stopped by a stop counter command. Note that this output is not masked by the contents of the IMR.
- The 1X clock for the channel B transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the channel B receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

OPCR[1:0] — OP2 Output Select — This field programs the OP2 output to provide one of the following:

- The complement of OPR[2]
- The 16X clock for the channel A transmitter. This is the clock selected by CSRA[3:0], and will be a 1X clock if CSRA[3:0] = 1111.
- The 1X clock for the channel A transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the channel A receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

Preliminary

ACR — Auxiliary Control Register

ACR[7] — Baud Rate Generator Set Select — This bit selects one of two sets of baud rates to be generated by the BRG:

Set 1: 50, 110, 134.5, 200, 300, 600, 1.05K, 1.2K, 2.4K, 4.8K, 7.2K, 9.6K, and 38.4K baud.

Set 2: 75, 110, 134.5, 150, 300, 600, 1.2K, 1.8K, 2.0K, 2.4K, 4.8K, 9.6K, and 19.2K baud.

The selected set of rates is available for use by the channel A and B receivers and transmitters as described in CSRA and CSRB. Baud rate generator characteristics are given in table 3.

ACR[6:4]—Counter/Timer Mode and Clock Source Select — This field selects the operating mode of the counter/timer and its clock source as shown in table 4.

ACR[3:0] — IP3, IP2, IP1, IPO Change of State Interrupt Enable — This field selects which bits of the Input Port Change register (IPCR) cause the input change bit in the interrupt status register (ISR[7]) to be set. If a bit is in the 'on' state, the setting of the corresponding bit in the IPCR will also result in the setting of ISR[7], which results in the generation of an interrupt output if IMR[7] = 1. If a bit is in the 'off' state, the setting of that bit in the IPCR has no effect on ISR[7].

IPCR — Input Port Change Register

IPCR[7:4] — IP3, IP2, IP1, IP0 Change of State — These bits are set when a change of state, as defined in the Input Port section of this data sheet, occurs at the respective input pins. They are cleared when the IPCR is read by the CPU. A read of the IPCR also clears ISR[7], the input change bit in the interrupt status register.

The setting of these bits can be programmed to generate an interrupt to the CPU.

IPCR[3:0] — IP3, IP2, IP1, IP0 Current State — These bits provide the current state of the respective inputs. The information is unlatched and reflects the state of the input pins at the time the IPCR is read.

ISR — Interrupt Status Register

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the reading of the ISR — the true status will be provided regardless of the contents of the IMR. The contents of this register are initialized to 00_{16} when the DUART is reset.

ISR[7] — Input Port Change Status — This bit is a '1' when a change of state has occurred at the IP0, IP1, IP2, or IP3 inputs and that event has been selected to cause an interrupt by the programming of ACR[3:0]. The bit is cleared when the CPU reads the IPCR.

ISR[6] — Channel B Change in Break — This bit, when set, indicates that the channel B receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel B 'reset break change interrupt' command.

ISR[5] — Channel B Receiver Ready or FIFO Full — The function of this bit is programmed by MR1B[6]. If programmed as receiver ready, it indicates that a character has been received in channel B and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel B FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

Table 3 BAUD RATE GENERATOR CHARACTERISTICS CRYSTAL OR CLOCK = 3.6864MHz

NOMINAL RATE (BAUD)	ACTUAL 16X CLOCK (KHz)	ERROR (PERCENT)
50	0.8	0
75	1.2	0
110	1.759	-0.069
134.5	2.153	0.059
150	2.4	0
200	3.2	0
300	4.8	0
600	9.6	0
1050	16.756	-0.260
1200	19.2	0
1800	28.8	0
2000	32.056	0.175
2400	38.4	0
4800	76.8	0
7200	115.2	0
9600	153.6	0
19.2K	307.2	0
38.4K	614.4	0

NOTE:
Duty cycle of 16X clock is 50% ± 1%.

Table 4 ACR [6:4] FIELD DEFINITION

ACR[6:4]	MODE	CLOCK SOURCE
0 0 0	Counter	External (IP2)[1]
0 0 1	Counter	TXCA — 1X clock of channel A transmitter
0 1 0	Counter	TXCB — 1X clock of channel B transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	External (IP2)[1]
1 0 1	Timer	External (IP2) divided by 16[1]
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

[1]In these modes, the channel B receiver clock should normally be generated from the baud rate generator.

Preliminary

ISR[4] — Channel B Transmitter Ready — This bit is a duplicate of TxRDYB (SRB[2]).

ISR[3] — Counter Ready — In the counter mode, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time that the counter/timer reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the counter/timer.

ISR[2] — Channel A Change in Break — This bit, when set, indicates that the channel A receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel A 'reset break change interrupt' command.

ISR[1] — Channel A Receiver Ready or FIFO Full — The function of this bit is programmed by MR1A[6]. If programmed as receiver ready, it indicates that a character has been received in channel A and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel A FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[0] — Channel A Transmitter Ready — This bit is a duplicate of TxRDYA (SRA[2]).

IMR — Interrupt Mask Register

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the programmable interrupt outputs OP3-OP7 or the reading of the ISR.

CTUR and CTLR — Counter/Timer Registers

The CTUR and CTLR hold the eight MSB's and eight LSB's respectively of the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded into the CTUR/CTLR registers is 0002_{16}. Note that these registers are write-only and cannot be read by the CPU.

In the timer (programmable divider) mode, the C/T generates a square wave with a period of twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half periods will be. In this mode the C/T runs continuously. Receipt of a start counter command (read with A3-A0 = 1110) causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR. The counter ready status bit (ISR[3]) is set once each cycle of the square wave. The bit is reset by a stop counter command (read with A3-A0 = 1111). The command, however, does not stop the C/T. The generated square wave is output on OP3 if it is programmed to be the C/T output.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR by the CPU. Counting begins upon receipt of a start counter command. Upon reaching terminal count (0000_{16}), the counter ready interrupt bit (ISR[3]) is set. The counter continues counting past the terminal count until stopped by the CPU. If OP3 is programmed to be the output of the C/T, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state and ISR[3] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter (CTU, CTL) may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8-bits to the upper 8-bits occurs between the times that both halves of the counter are read. However, note that a subsequent start counter command will cause the counter to begin a new count cycle using the values in CTUR and CTLR.

IVR — Interrupt Vector Register

This register contains the interrupt vector. The register is initialized to H'0F' by RESET. The contents of the register are placed on the data bus when the DUART responds to a valid interrupt acknowledge cycle.

Preliminary

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to + 70	°C
Storage temperature	− 65 to + 150	°C
All voltages with respect to ground[3]	− 0.5 to + 6.0	V

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+ 70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage (except X1/CLK)		2.0			V
V_{IH}	Input high voltage (X1/CLK)		4.0			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except o.c. outputs)	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	− 10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_O = 0$ to V_{CC}	− 10		10	μA
I_{OC}	Open collector output leakage current	$V_O = 0$ to V_{CC}	− 10		10	μA
I_{CC}	Power supply current				150	mA

NOTES:
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at + 25°C, typical supply voltages, and typical processing parameters.

Preliminary

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$ [4, 5, 6, 7]

PARAMETER		TENTATIVE LIMITS			UNIT
		Min	Typ	Max	
Reset Timing (figure 1)					
t_{RES}	RESETN pulse width	1.0			μs
Bus Timing (figures 2, 3, 4)					
t_{AS}	A1-A4 setup time to CSN low	10			ns
t_{AH}	A1-A4 hold time from CSN low	0			ns
t_{RWS}	RWN setup time to CSN low	0			ns
t_{RWH}	RWN holdup time to CSN high	0			ns
t_{CSW}[8]	CSN high pulse width	160			ns
t_{CSD}[9]	CSN or IACKN high from DTACKN low	20			ns
t_{DD}	Data valid from CSN or IACKN low			175	ns
t_{DF}	Data bus floating from CSN or IACKN high			100	ns
t_{DS}	Data setup time to CLK high	100			ns
t_{DH}	Data hold time from CSN high	0			ns
t_{DAL}	DTACKN low from read data valid	0			ns
t_{DCR}	DTACKN low (read cycle) from CLK high			125	ns
t_{DCW}	DTACKN low (write cycle) from CLK high			125	ns
t_{DAH}	DTACKN high from CSN or IACKN high			100	ns
t_{DAT}	DTACKN high impedance from CSN or IACKN high			125	ns
t_{CSC}[10]	CSN or IACKN setup time to clock high	90			ns
Port Timing (figure 5)					
t_{PS}	Port input setup time to RDN low	0			ns
t_{PH}	Port input hold time from RDN high	0			ns
t_{PD}	Port output valid from WRN high			400	ns
Interrupt Reset Timing (figure 6)					
t_{IR}	INTRN, or OP3-OP7 when used as interrupts, high from:				
	Read RHR (RxRDY/FFULL interrupt)			300	ns
	Write THR (TxRDY interrupt)			300	ns
	Reset command (delta break interrupt)			300	ns
	Stop C/T command (counter interrupt)			300	ns
	Read IPCR (input port change interrupt)			300	ns
	Write IMR (clear of interrupt mask bit)			300	ns
Clock Timing (figure 7)					
t_{CLK}	X1/CLK high or low time	100			ns
f_{CLK}	X1/CLK frequency	2.0	3.6864	4.0	MHz
t_{CTC}	CTCLK high or low time	100			ns
f_{CTC}	CTCLK frequency	0		4.0	MHz
t_{RX}	RXC high or low time	220			ns
f_{RX}	RXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
t_{TX}	TXC high or low time	220			ns
f_{TX}	TXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
Transmitter Timing (figure 8)					
t_{TXD}	TXD output delay from TXC low			350	ns
t_{TCS}	TXC output skew from TXD output data			150	ns
Receiver Timing (figure 9)					
t_{RXS}	RXD data setup time to RXC high	240			ns
t_{RXH}	RXD data hold time from RXC high	200			ns

NOTES:

4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: $C_L = 150pF$, except interrupt outputs. Test condition for interrupt outputs: $C_L = 50pF$, $R_L = 2.7K$ ohm to V_{CC}.
8. This specification will impose maximum 68000 CPU CLK to 6MHz. Higher CPU CLK can be used if repeating bus reads are not performed.
9. This specification imposes a lower bound on CSN and IACKN low, guaranteeing that it will be low for at least 1 CLK period.
10. This specification is made only to insure that DTACKN is asserted with respect to the rising edge of the X1/CLK pin as shown in the timing diagram, not to guarantee operation of the part. If the setup time is violated, DTACKN may be asserted as shown, or may be asserted one clock cycle later.

Preliminary

Figure 1. Reset Timing

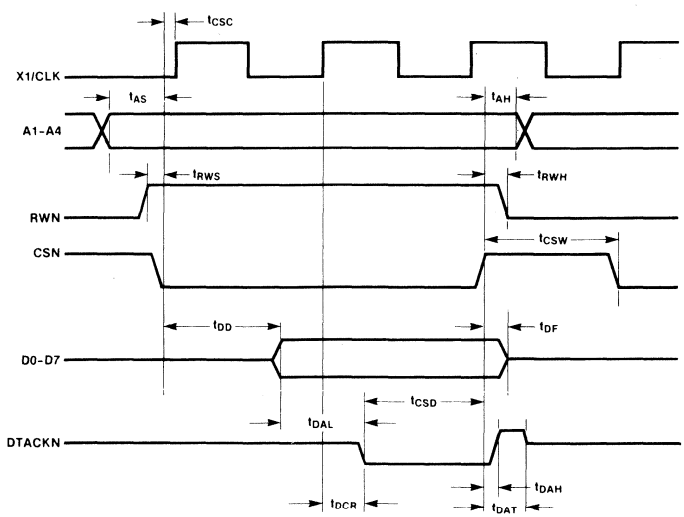


Figure 2. Bus Timing (Read Cycle)

Preliminary

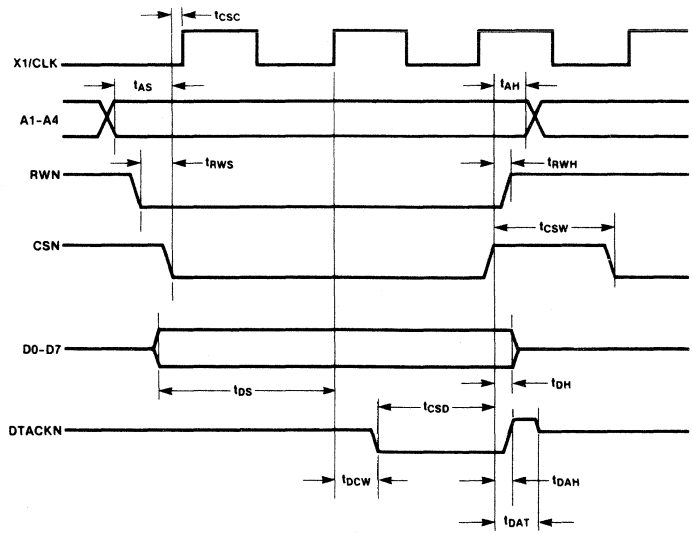


Figure 3. Bus Timing (Write Cycle)

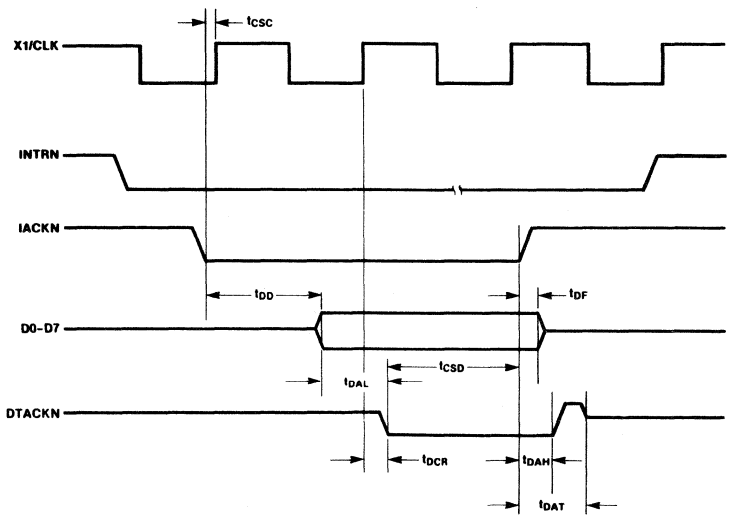


Figure 4. Interrupt Cycle Timing

Preliminary

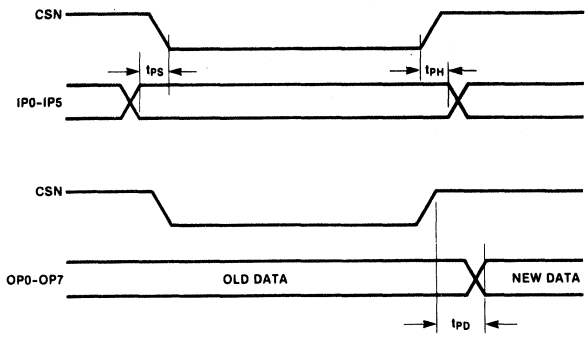
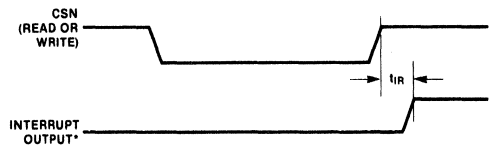


Figure 5. Port Timing



*INTRN or OP3-OP7 when used as interrupt outputs.

Figure 6. Interrupt Timing

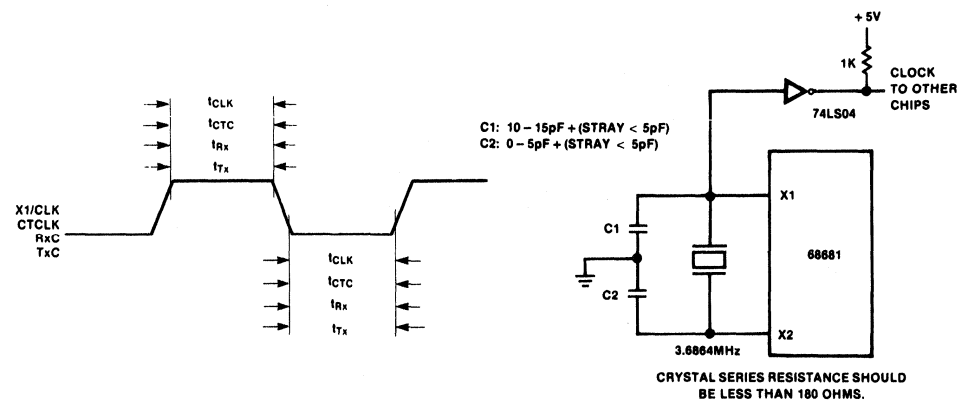


Figure 7. Clock Timing

Preliminary

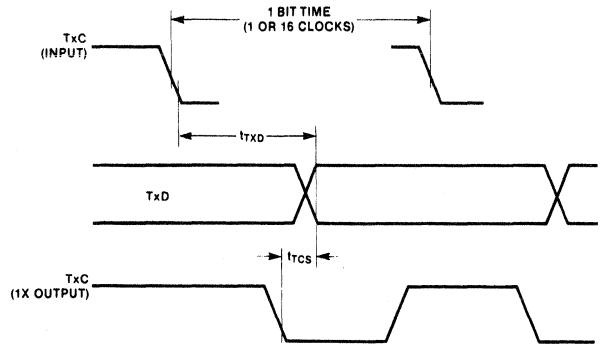


Figure 8. Transmit Timing

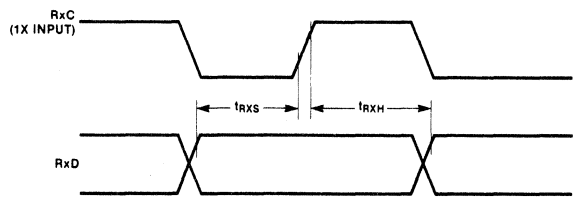
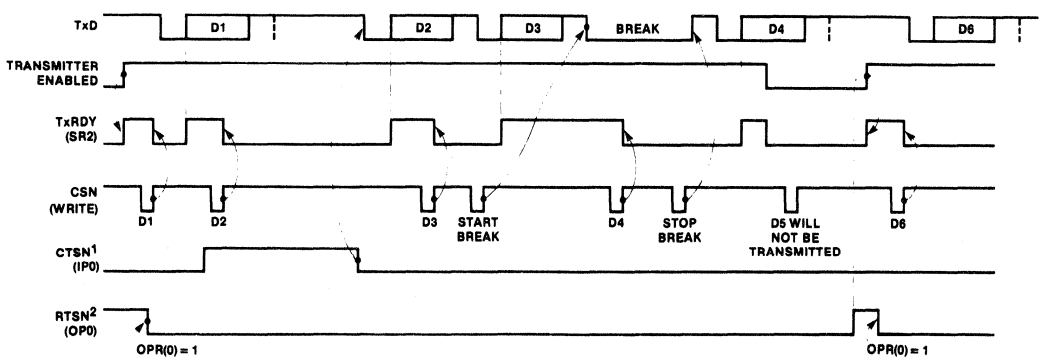


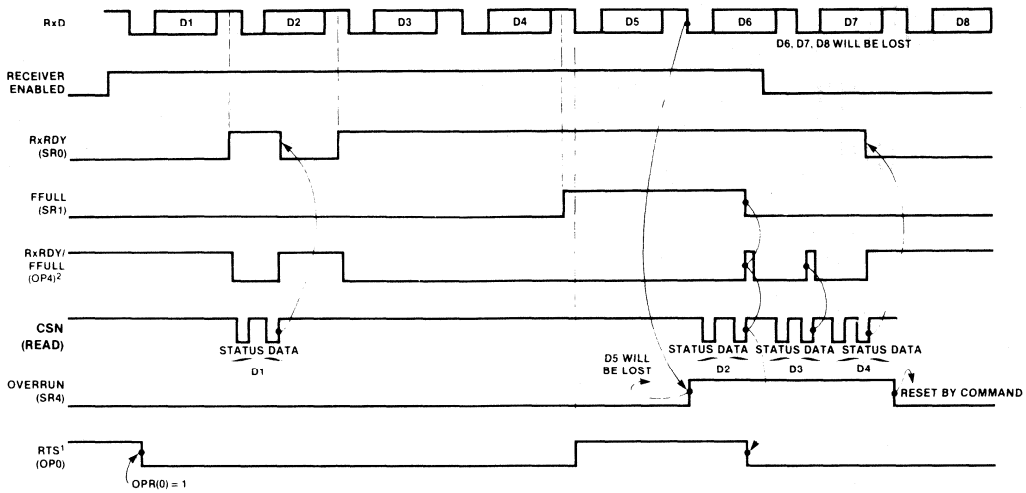
Figure 9. Receive Timing



NOTES
1. TIMING SHOWN FOR MR2(4) = 1.
2. TIMING SHOWN FOR MR2(5) = 1.

Figure 10. Transmitter Timing

Preliminary



NOTES
1. TIMING SHOWN FOR MR1(7) = 1.
2. SHOWN FOR OPCR(4) = 1 AND MR1(6) = 0.

Figure 11. Receiver Timing

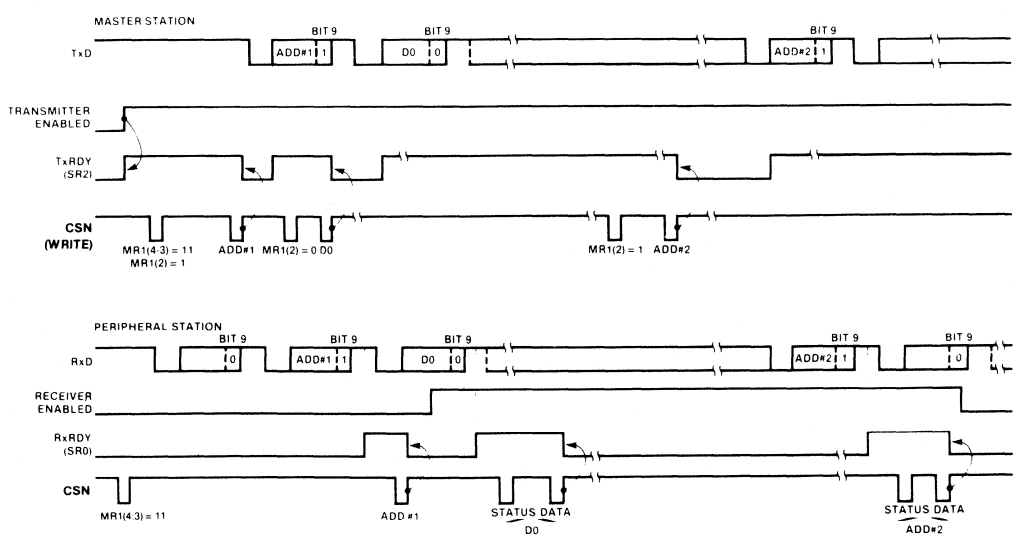


Figure 12. Wake up Mode

DESIGN ENTRY

Fast data-comm controller speaks to all protocols over two sets of channels

A data-communication controller IC gets its message across dual receiver and transmitter channels at 4 Mbits/s. It has yet to meet a protocol it does not like.

As a diversity of computers work their way into ever more factories and offices, the pressure is steadily building for an easy way to share and exchange information among them. At odds with this simple idea is a tangle of different data-communication protocols, each tailored to serve a specific need and demanding strict adherence to its own set of rules. Any previous attempts to untangle communication, using a circuit that handles multiple protocols and their variations, has demanded too much hardware.

Finally cutting through this barrier is the SCN68562 dual universal serial communications controller. The chip is designed to be the foundation of a universal high-performance data-communication subsystem, particularly for the 68000 family of microprocessors. At 4 Mbits/s, the controller's data rate is nearly twice that currently needed in even the most powerful systems. At the same time, its two independent pairs of transmitter and receiver channels are easily and separately configured to handle a wide range of protocols. Each channel can operate in full-duplex synchronous or asynchronous modes and can adhere to the following character- and bit-oriented protocols: HDLC/ADCCP, SDLC, IBM's Bisync, DDCMP, X.21, X.25, and X.75 (at the link level).

Once configured for a particular protocol, the chip operates with minimal intervention—usually just a few commands—from its host processor. This is because the controller carries so much of its own interface hardware, as well as a wealth of operational and status registers. For example, it contains all the circuitry needed to interface with advanced direct-memory access (DMA) controllers. In addition, it has numerous features for interrupting the host processor, including vectored interrupts, daisy chaining, masking, and prioritization.

Timing, counting, and generating the proper clock signals come naturally to the controller because it commits a 16-bit timer and counter and a digital phase-locked loop to each channel, as well as sharing a crystal oscillator and baud-rate generator between them.

Another way in which the controller offloads its host is with its ability to handle special control characters automatically. Each trans-

Jim Magill and **George Wong,** Signetics Corp.

Jim Magill works at Signetics's Microprocessor Division in Sunnyvale, Calif., where he is a product marketing manager for 68000 family and data communications products. He is a member of the Institute of Production Engineers, and holds a BS from Queens University in Belfast and an MS from City University in London.

George Wong is a senior applications engineer at the same division. He has been at Signetics for five years and has previously worked at AM Varityper, IBM, Instrumentation Engineering and General Electric. He earned his BSEE and MSEE degrees at the University of Louisville.

Reprinted from **ELECTRONIC DESIGN** - January 24, 1985

mitter can generate, as each receiver can accept, special character- and error-checking sequences often used in advanced message formats.

Four major functional sections make up the controller's architecture. They are the interface and operational controls, the timing section, the channel receivers, and finally the channel transmitters (Fig. 1).

The first, the interface and operational control section, handles transactions between the controller and a host processor, interrupting devices, and direct-memory-access controller. To this end, the section contains a multitude of registers for configuring the controller's operation (Table 1).

Among these registers are two Channel-mode Configuration Registers (CMR_1 and CMR_2). Each selects its channel's protocol and transmission mode, message format, and error-checking sequence, as well as the overall system interface. Another register, the Pin Configuration Register (PCR), programs the chip's multifunction pins. The Interrupt Control Register (ICR) and the Interrupt Enable Register (IER) determine how the controller will interrupt the host processor. The first also selects which channel's interrupts have higher priority, and whether to issue vectored or nonvectored interrupts.

Interrupt variety

In the vectored interrupt mode the chip issues either a fixed base vector or one that is encoded with status information. The controller puts the vector on the system data bus during the interrupt-acknowledgment cycle, when the processor expects it. In all, the controller can generate up to eight different vectors. Interrupt masking can be set for an entire channel or limited to a group of channel conditions, using ICR and IER, respectively.

Just three registers for each channel indicate the controller's status with enough detail and precision for the most advanced data communications system. They are the Receiver Status Register (RSR), the Transmitter and Receiver Status Register (TRSR), and the Input Port, Counter, and Timer Status Register (ICTSR).

The exact status information stored in the first two registers depends on the channel's protocol and transmission modes. The ICTSR contains the status of the counter and timer, as well as that of the input port pins. These pins connect to the chip's logic circuits that detect a status change.

A General Status Register (GSR) holds a summary of the status of both channels and gives a quick indication of the their overall state. Finally there is a Channel Command Register for each channel (CCRA and CCRB), which holds commands issued to the chip (Table 2).

Timing untangled

The chip's next major functional section is embodied in its timing generation circuit. Using a crystal oscillator, this section generates the multitude of different timing and control signals needed for a baud-rate generator and contains, for each channel, a digital phase-locked loop (DPLL) and a 16-bit timer and counter.

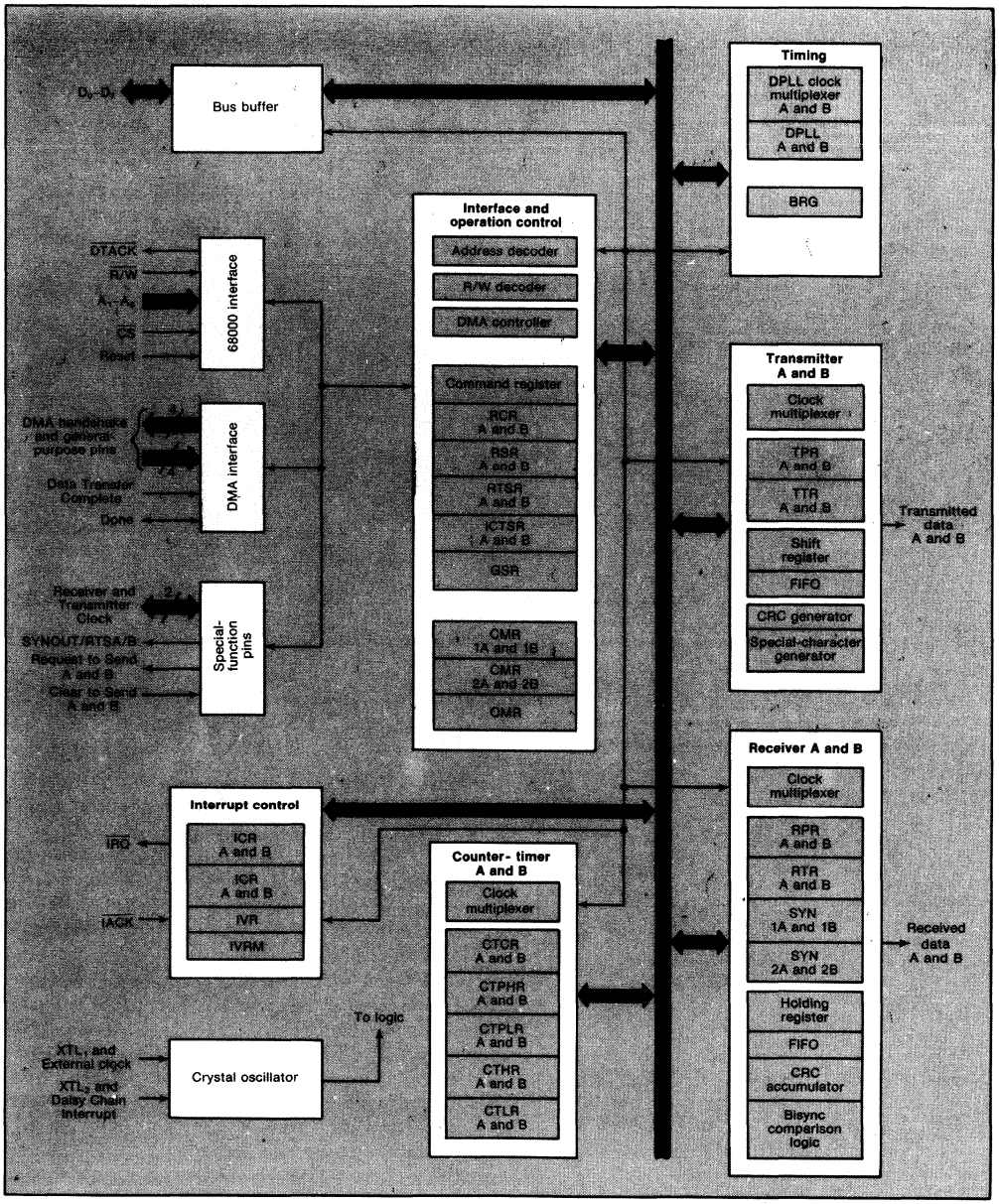
The crystal oscillator operates directly from a crystal or an external clock source. The oscillator's output drives the controller's internal logic and timing circuits, like the baud-rate generator.

The baud-rate generator produces 16 rates at once, which are available to the receiver, transmitter, digital phase-locked loop, and counter-timer. Having so many simultaneous baud rates allows rates for each receiver and transmitter to be independently set.

The digital phase-locked loop for each channel recovers a data clock from a received synchronous data stream. The data clock, in turn, can be used in the receiver or transmitter sections. Permissible synchronous encoding schemes include nonreturn to zero (NRZ) and nonreturn to zero inverted (NRZI), as well as Manchester and two other forms of frequency modulation, FM_0 and FM_1 (Fig. 2).

In addition, a sampling clock, required by the digital phase-locked loop, can come from a number of sources, including an external input, the receiver's baud-rate generator, the counter-timer, or the chip's crystal oscillator.

The next section, the counter-timer, can serve as a programmable divider, periodic interrupt generator, delay and bit-length timer, and character and event counter. This section is



1. The architecture of a new 4-Mbit/s two-channel universal serial communications controller consists of interface and operational controls, a timing section, and two receivers and two transmitters. A healthy array of registers configures the controller and helps to offload a host processor.

sufficiently powerful to require only an input to its crystal oscillator to meet the counting and timing functions of most applications.

The counter and timer consists of a control register (CTCR), a 16-bit down counter, a 16-bit preset register, an input-clock multiplexer, and associated control logic. The control register governs this section by selecting its clock source, prescaling factor, and operating mode.

Table 1. Control registers

Acronym	Name
CMR1	Channel mode register 1
CMR2	Channel mode register 2
SYN1	Secondary address 1 register
SYN2	Secondary address 2 register
TPR	Transmitter parameter register
TTR	Transmitter timing register
RPR	Receiver parameter register
RTR	Receiver timing register
CTPRH	Counter-timer preset register high
CTPRL	Counter-timer preset register low
CTCR	Counter-timer control register
OMR	Output and miscellaneous register
CTH	Counter-timer high
CTL	Counter-timer low
PCR	Pin configuration register
CCR	Channel command register
TXFIFO	Transmitter FIFO
RXFIFO	Receiver FIFO
RSR	Receiver status register
TRSR	Transmitter and receiver status register
ICTSR	Input and counter-timer status register
GSR	General status register
IER	Interrupt enable register
IVR	Interrupt vector register (unmodified)
IVRM	Interrupt vector register (modified)
ICR	Interrupt control register

Table 2. Channel command register

Bits 7–6	Bits 5–4	Bits 3–0
00 Transmitter commands	Don't care	0000—reset TX 0001—reset TX CRC 0010—enable TX 0011—disable TX 0100—send SOM 0101—send SOM with PAD 0110—send EOM 0111—send Break/Abort 1000—send DLE 1001—reserved 1010—reserved 1011—reserved 1100—reserved 1101—exclude from CRC
01 Receiver commands	Don't care	0000—reset RX 0001—reserved 0010—enable RX 0011—disable RX
10 Counter-timer commands	Don't care	0000—start 0001—stop 0010—preset to FFFF 0011—preset from CTPRH/CTPRL
11 Digital phase-locked loop commands	Don't care	0000—enter search mode 0001—disable PLL 0010—set FM mode 0011—set NRZI mode 0100—reserved for test 0101—reserved for test

Input clock sources to the multiplexer include an external source, crystal oscillator, receiver and transmitter baud-rates, and internal signals that transfer data between the chip's FIFOs and shift registers. The clock can be controlled by commands, countdown conditions, or even the logic state of a receiver's data line, and can be prescaled by factors of 64, 32, and 16.

The transmitter is the third of the chip's major functional sections. Each channel contains a completely independent transmitter, which in turn comprise three primary subsections. The subsections consist of control registers and a transmitter clock multiplexer, a shift register (TXSR) and transmitter FIFO, and special-character generation logic.

After the protocol and transmission mode of a channel is selected, the transmitter's operation is further defined by the control registers: the Transmitter Parameter Register (TPR) and the Transmitter Timing Register (TTR). The effect of the former depends on the selected protocol and transmission mode.

The clock multiplexer selects the transmitter's clock source from among the counter-timer sections of either channel, the baud-rate generator, the digital phase-locked loop, and an external clock source. When the baud-rate generator is selected, the Transmitter Timing Register chooses the baud rate.

The transmitter accepts data from the system data bus into its FIFO (TXFIFO), which consists of four 8-bit holding registers. From there, data moves to the Transmitter Shift Register (TXSR), which serializes it for the selected format. Cyclic- and longitudinal-redundancy check (CRC and LRC) characters can also be loaded into the shift register from the chip's special character logic.

When the FIFO is ready to accept data, it indicates that by setting a Transmit Ready status bit. The point at which the FIFO is ready is controlled by the user, through one of the two Output and Miscellaneous Registers (OMRA or OMRB), depending on the channel in question. The choice is between a completely empty FIFO or an empty position within the FIFO as the point at which to set the Transmit Ready status bit. Because the DMA and interrupt service requests also adhere to the state of the Transmit Ready bit, the user can select how often ser-

vice requests are generated.

The last subsection within each transmitter is composed of special logic to calculate CRC-16 and CRC-CCITT or longitudinal-redundancy-check characters. Either of these block control characters is generated by accumulating the character data sent to the transmitter's serial register. In all, the designer has a choice of six, rather than three, different block-control characters, because the polynomial for generating the CRC and LRC characters can be initialized to all 1s and all 0s.

This subsection also executes commands to exclude a character from CRC accumulation, send the block-control character as the end-of-message character, or preset and reset the register that holds the block-control character.

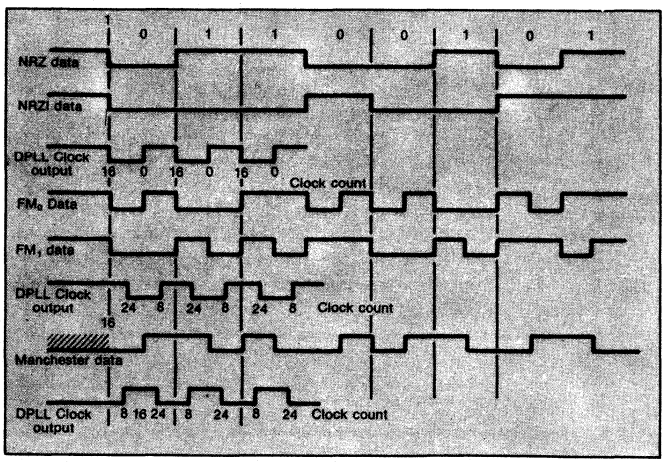
Other circuitry within this subsection will transmit special bit patterns for the various transmission modes (Table 3). For example, for character-oriented protocols, it will send a synchronization pattern or a data link enable character (for bisynchronous operation). For bit-oriented protocols, the chip can send a flag (01111110) at the start and end of a message, as well as an abort pattern (11111111). It also can send a character of all 0s, signifying a break in an asynchronous protocol.

The last major functional section of the data controller chip is the receiver. Each of the two independent receivers has four subsections. These are grouped as control registers and a clock multiplexer, FIFO and shift register, data path, and error accumulation logic.

For control, the receiver looks to the Receiver Parameter and Receiver Timing Registers (RPR and RTR). The exact options set by the former depend on the selected transmission mode.

The timing register selects the clocking source and, if that source is the baud-rate generator, what the rate will be. It also selects the source for the digital phase-locked loop, which may be the counter-timer, an external input, the baud-rate generator, or the crystal oscillator. The receiver timing multiplexer picks as the timing signal source either an external clock, the counter-timer, baud-rate generator, or the digital phase-locked loop.

The next receiver subsection, the data path, is best viewed as four separate subpaths: one for asynchronous and three for variations of the synchronous protocols: bit-oriented with and without CRC, and character-oriented with bi-



2. The controller dedicates a built-in digital phase-locked loop (DPLL) to each of its two channels. The phase-locked loop serves to recover clock pulses from a received data stream, giving the controller its power to manipulate an assortment of synchronous encoding schemes, including NRZ and NRZI, Manchester encoding, and two other forms of frequency modulation called FM_0 and FM_1.

synchronous protocols.

Each data path assembles characters into a receiver shift register and sends them to the FIFO, where any required status information is appended. The FIFO consists of four holding registers, each with eight data bits and status information. As in the transmitter section, a user can select when the ready-status bit is set, or when the FIFO is either completely or partially full. Because the DMA and interrupt requests respond to the state of the ready bit, this option for setting the ready status can also control bus traffic.

Significantly, the controller contains comparison logic for processing text in the bisync transmission mode. The logic can identify specific EBCDIC and ASCII characters and sequences of characters without help from the host processor. When the receiver identifies characters like Start of Header (SOH) and Start of Text and character sequences like Temporary Text Delay (TTD), it can set bits in the Receiver Status Register or initiate special processing. This feature is active for normal and transparent bisynchronous modes.

Because the controller, as the heart of an advanced communications system, includes so much of the required hardware, very little extra is needed to build, for example, a full-duplex channel that uses DMA data transfers (Fig. 3).

The DMA mechanism takes advantage of the chip's separate request and acknowledge signals for the transmitter and receiver. The protocol is bit-oriented and uses a secondary address mode, a single control byte, and a CCITT 16-block character check that is preset to all 0s. The transmitter clock source is supplied externally, and the digital phase-locked loop derives the receiver clock from the Manchester encoding and an internal sampling clock. For transferring data to and from the host processor, the Transmit Ready bit is asserted when the transmitter FIFO is empty and the Receiver Ready bit is asserted when the receiver FIFO is full.

Because the controller contains its own digital phase-locked loop, baud-rate generator, and crystal oscillator, the only external timing device required is the crystal. From it, the necessary timing relationships are programmed from the channel's Receiver and Transmitter Timing Registers and the Pin Control Register.

Table 3. Built-in characters

Acronym	Meaning	
SOH	Start of header	Single-character sequences (normal text)
STX	Start of text	
ETX	End of text	
EOT	End of transmission	
ENQ	Enquiry	
DLE	Data link escape	
NAK	Negative acknowledge	
ETB	End of block	
ITB	Intermediate block	
ACK0	Acknowledge 0	Two-character sequences (normal text)
ACK1	Acknowledge 1	
WACK	Wait before transmitting ACK	
RVI	Reverse interrupt	
TTD	Temporary text delay	
DLE ENQ	Enquiry	Transparent mode
DLE ITB	Intermediate block	
DLE ETB	End of block	
DLE ETX	End of text	
DLE STX	Transparent text mode	

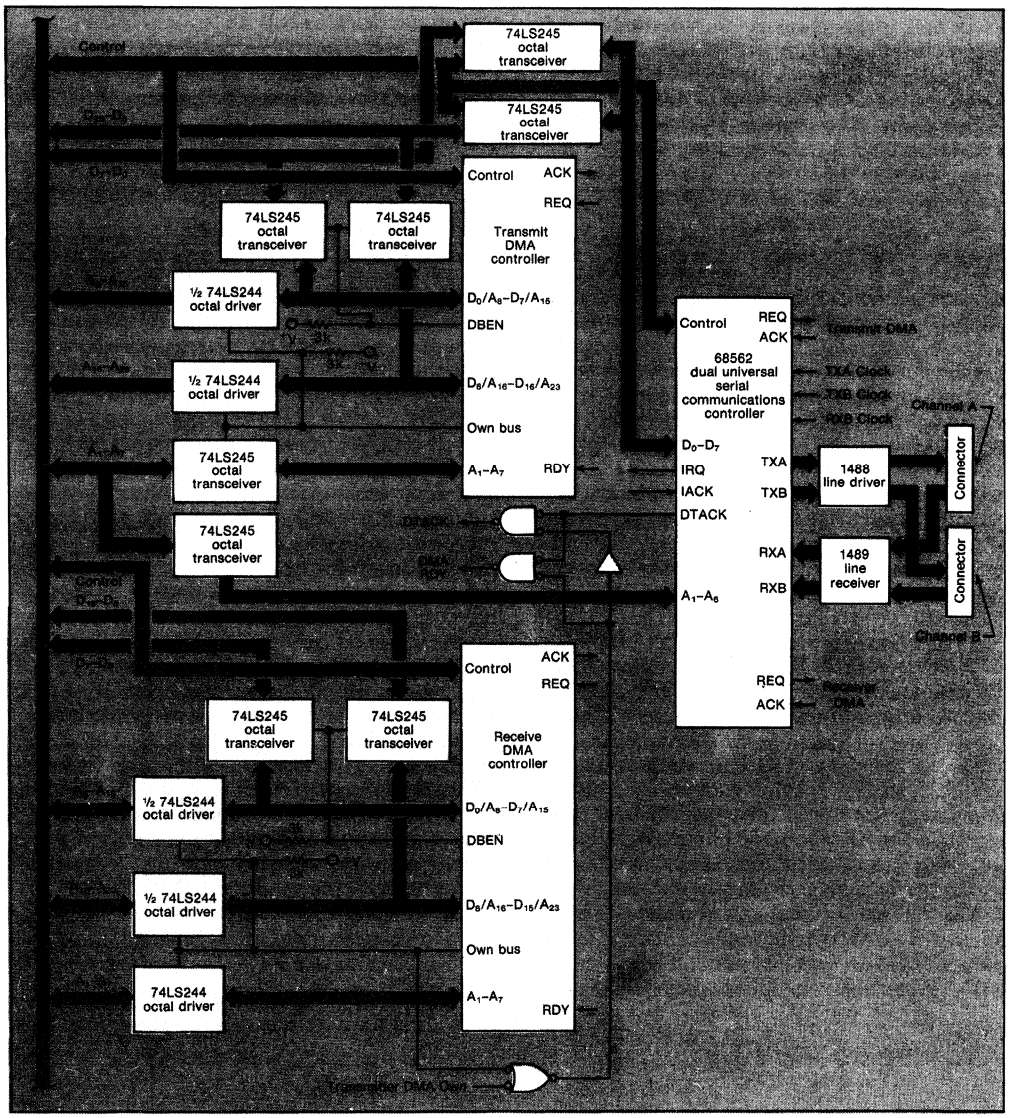
Simple software

Just as the hardware making up this system is minimal, so is the initialization routine for configuring the interface (see the program, p. 166). Specifically, five instructions, each loading one register, initialize the channel: Channel configuration registers CMR_1 and CMR_2 establish the DMA structure and define the data channel's operation; two instructions store the address of the secondary station in secondary address registers SYN_1 and SYN_2; and the external inputs are defined by one instruction to load the Pin Control Register.

The Transmitter and Receiver Parameter Registers further define the channel's operation by selecting the message format and various automatic functions. For example, the transmitter, automatically and without intervention by the host processor, terminates the message by sending an End of Message sequence and marking the data line (sending marks, or 1s) when an underrun occurs.

To configure the chip to a completely different set of system specifications requires only changing a few bits in the initialization registers. In whatever way the chip's interface is configured, once this is done the transmitter and receiver are controlled and monitored by the Channel Command Register and the Transmitter and Receiver Status Register.

In this example, six commands come into play for transmitting a bit-oriented-protocol



3. As the heart of this advanced communications system, the new controller chip needs little extra hardware. The system uses a bit-oriented protocol with Manchester encoding and takes advantage of the chip's DMA interface for transferring data. Because the controller contains its own digital phase-locked loop, the only external timing device required is a crystal.

(BOP) message. With them, a processor can transmit frames or invoke automatic features in a straightforward manner and with a minimum of intervening hardware, even when sending special patterns. For the processor, sending a BOP message reduces to the simple task of loading the FIFO memory.

For example, the Transmit Start of Message (TSOM) command starts the message by first initializing the transmitter CRC and sending a flag pattern (01111110). A command, TSOMP, does the same, but precedes the flag with a padding sequence for synchronizing a digital phase-locked loop at the receiving end of the message. A variable number of address bytes, plus one or two control bytes, are loaded into the transmitter FIFO following the flag pattern.

After the last control byte is sent, the transmitter automatically switches to its programmed character length for sending the rest of the message. The other commands send an End of Message character; abort the transmission by clearing the transmitter FIFO and sending an Abort character; reset the CRC; and exclude a character from the CRC.

The Transmitter Parameter Register allows the user to select the underrun termination and idle sequences of the message frame. The six choices are: sending a normal ending sequence (in which the block check character is followed by a flag) with or without continuous marks or continuous flags or sending an abnormal ending sequence (in which the block check character is omitted) with or without continuous marks or continuous flags.

Also, the End of Message sequence can be explicitly ordered by a command that appends the sequence to the next character loaded into the FIFO. Alternatively, the sequence can be invoked automatically at the end of a DMA transfer. It can be invoked by the counter-timer section when the latter is counting characters and detects the last one is being sent.

One of the transmitter's most powerful features using a bit-oriented protocol is its ability to transmit a final character that has fewer bits than the programmed character length. When a shorter character length is specified in the Output and Miscellaneous Register, the last character of the End of Message sequence is transmitted with the shortened number of bits.

Finally, the transmitter section automatically inserts 0s into a data stream to prevent mistakenly transmitting the flag sequence. Regardless of any character boundaries, it inserts a 0 in the serial data stream whenever five consecutive 1s have been transmitted. All characters loaded into the Transmitter Serial Register from the FIFO as well as those from the CRC generator are subject to the automatic zero-insertion feature.

The chip's receiver, in the bit-oriented protocol mode, has four phases that it switches among: hunt, address field, control field, and information field. Because the receiver handles all the overhead tasks and indicates all of the message conditions for each of the phases, the host processor's intervention after initialization is almost trivial.

The receiver first enters the hunt phase, in which it performs a comparison every bit time until a flag sequence is identified. The flag sequence marks the beginning and end of a received frame and establishes a character boundary. A match sets the Flag Detection status bit, but the sequence itself is deleted from the data stream.

After the receiver detects a flag, it enters the address phase. The length of an anticipated ad-

Getting Ready

```
; Channel A initialize
    MOVE.B  #$19,CMR1A  ;Manchester, encoding, BOP secondary mode
    MOVE.B  #$14,CMR2A  ;Full duplex DMA, CRC-16, preset to 0s
    MOVE.B  ADD1,SYN1A  ;High part of station address
    MOVE.B  ADD2,SYN2A  ;Low part of station address
    MOVE.B  #$00,PCRA   ;External transmit pin
; Transmitter Parameters
    MOVE.B  #$01,TPRA   ;Idle in mark, Normal EOM if underrun
    MOVE.B  #$80,TTRA   ;Select external transmit clock
; Receiver parameters
    MOVE.B  #$08,RPRA   ;All parties wake-up mode
    MOVE.B  #$40,RTRA   ;Receiver clock DPLL, crystal driven
; TXRDY and RXRDY
    MOVE.B  #$18,OMRA   ;TXRDY when FIFO is empty, RXRDY when FIFO is full
```

dress and the way the receiver handles it depends on how the Channel-Mode configuration register, which selects the secondary address modes, is programmed. If an address match occurs or the received character sequence indicates a group or all-parties address, the chip loads all the characters of the address, control, and information fields into the receiver FIFO. The character lengths for the address and control fields are eight bits, and the character length for the information field is set by the Receiver Parameter Register.

The control field phase follows the address field phase. In it the receiver accepts one or two control characters, depending on how the Channel-Mode configuration Register is programmed. After that, the character length automatically switches from eight bits to the number specified in the Receiver Parameter Register, making the receiver enter the information field phase. There it accepts the message-frame data. When the receiver detects the next flag sequence, it terminates the frame. The same flag, however, can also serve as the start of the next frame.

The receiver assumes that the 16 bits received prior to the closing flag are the block character check, provided a check was specified. All frame characters, except the flag, are accumulated in the CRC checker and the result is compared with an expected value. A failure to compare causes a CRC error, which, like other status information, is stored in the FIFO after the last character of the information field.

The receiver also indicates a short frame error if a closing flag arrives before the arrival of the proper number of address fields, control fields, and block check character octets. It also takes note if the message is aborted or if an idle sequence follows the opening flag. Finally, to compensate for the transmitter's zero-insertion feature, the receiver deletes a 0 after five contiguous 1s. Flag, abort, and idle sequences are all done before any 0s are deleted. □

Interface

VMEbus Controller (BUSCON)

Originally published by Signetics February 1985

Advance Information

DESCRIPTION

The Signetics SCB68172 VMEbus Controller (BUSCON) is an interface device for the VMEbus. It can be used in three different configurations: master-only, slave-only, and master/slave. The SCB68172 can be used with a processor-type interface or with a DMA controller-type interface. In all configurations, it handles the VMEbus signaling protocol in compliance with revisions B and C of the VMEbus Specification.

CONFIGURATION/VERSION

Applications of the BUSCON are identified as follows (see figures 1 through 4):

VERSION	APPLICATION
PMS	Processor-type master/slave
DMAC	DMA controller-type master/slave
MS	Either PMS or (DMAC)
M	Master-only
S	Slave-only

All of these applications are handled by the SCB68172, with unused pins tied to stated logic levels in some of the applications.

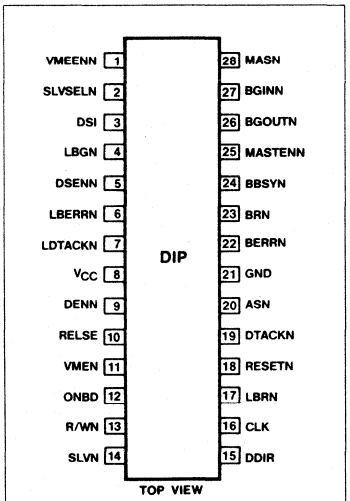
Figure 5 shows a functional model of the SCB68172 logic. The ASN, MASN, LBRN, BGINN, and RELSE inputs are internally synchronized to CLK before being presented to the state machine which determines the major functions of the device. The SLVN, ONBD, and VMEN signals are used directly in the state machine, although they are highly qualified to prevent metastable conditions on the state machine outputs. The BRN, BBSYN and LBGN signals are direct state machine outputs, while ASN, MASTENN, VMEENN, SLVSELN, and BGOUTN are derived from the state machine outputs plus some combinatorial qualification.

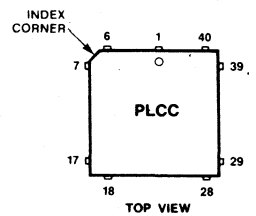
The DSI, R/WN, DTACKN, BERRN, LDTACKN, and LBERRN inputs function largely as direct combinatorial inputs. The DDIR, DTACKN, BERRN, LDTACKN, LBERRN, and (when applicable) MASN outputs are largely derived directly from these direct inputs, with some qualification from the state machine outputs. The DENN and DSENN outputs have complex multi-case logic which uses both the direct inputs and the state machine outputs.

FEATURES

- **Master, slave, or master/slave (dual ported) applications**
- **Helps assure VMEbus compatibility**
- **Allows for address decoding time**
- **Processor or DMA controller interface for master/requester**
- **Master/requester logic allows release on request (ROR) or release when done (RWD) operation, early or intercycle release**
- **Supports and exploits address lookahead**

PIN CONFIGURATION





Pin	Function	Pin	Function
1	NC	23	NC
2	VMEENN	24	DDIR
3	SLVSELN	25	CLK
4	DSI	26	LBRN
5	LBGN	27	RESETN
6	NC	28	NC
7	NC	29	NC
8	DSENN	30	DTACKN
9	LBERRN	31	ASN
10	NC	32	NC
11	LDTACKN	33	GND
12	NC	34	NC
13	V_{CC}	35	BERRN
14	NC	36	NC
15	DENN	37	BRN
16	RELSE	38	BBSYN
17	NC	39	NC
18	NC	40	NC
19	VMEN	41	MASTENN
20	ONBD	42	BGOUTN
21	R/WN	43	BGINN
22	SLVN	44	MASN

Advance Information

ORDERING CODE

PACKAGES	V_{CC} = 5V±5%, T_A = 0°C to 70°C
Ceramic DIP	SCB68172AC25I28
Plastic DIP	SCB68172AC25N28
Plastic LCC	SCB68172AC25A44

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	CONFIG	NAME AND FUNCTION
	DIP	PLCC			
CLK	16	25	I	All	**Clock:** User-supplied clock signal.
SLVN	14	22	I	S,MS	**Slave:** Active-low decode of the VMEbus address and address modifier lines indicating that the current cycle is for this board. SLVN should not be qualified with ASN nor VMEENN. It is first sampled on the rising clock edge after the rising edge on which ASN is first detected. It must remain valid until after the next low-going edge on DTACKN or BERRN. In a master-only application, SLVN should be pulled up to V_{CC}.
ASN	20	31	I/O I	M,MS S	**Address Strobe:** Direct connect to VMEbus ASN.
VMEN	11	19	I	M,MS	**VME Decode:** Active-low decode of the master's address lines, indicating that the master's current cycle is for a slave on the VMEbus. VMEN should not be qualified with MASN nor MASTENN. It is first sampled on the rising clock edge after the one on which MASN is first detected. Thereafter, it must remain valid until MASN goes false (high). In a slave-only configuration, VMEN should be pulled up to V_{CC}.
LBRN	17	26	I	M,MS	**Local Bus Request:** Connected to the low-active bus request output of a DMA controller. Typically tied to a high logic level in processor-type interfaces.
ONBD	12	20	I	MS	**Onboard:** Active-high decode of the master's address lines, indicating that the master's current cycle is for an onboard slave that is dual-ported with the VMEbus. ONBD should not be qualified with MASN or MASTENN. It is first sampled on the rising clock edge after the one on which MASN is first detected. Thereafter, it must remain valid until after MASN goes false (high). In a master-only or slave-only application, ONBD should be grounded. If a master/slave configuration does not contain "local slaves" as shown in figure 3, VMEN and ONBD should both be connected to an active-low "VME decode". A cycle between the onboard master and a local slave (VMEN high, ONBD low) is ignored by BUSCON, and can proceed concurrently with a cycle between another VMEbus master and an onboard dual-ported slave.
MASN	28	44	I I/O	M,PMS DMAC	**Master's Address Strobe:** RMW and Sequential VMEbus master cycles are accomplished by holding MASN low across several data strobes. If LBGN is high at the end of the RESETN low time, the state of ASN is driven onto MASN whenever BUSCON does not have control of the VMEbus. In a slave-only application, MASN should be pulled up to V_{CC}.
MASTENN	25	41	O	MS	**Master Enable:** In a master/slave application, the low state of this signal enables the master onto the shared bus and enables shared-bus responses back to the master. MASTENN also provides the direction control for the VMEbus address transceivers.
VMEENN	1	2	O	M,MS	**VME Enable:** Active-low enable for the VMEbus address drivers (master-only) or transceivers (master/slave).
SLVSELN	2	3	O	S,MS	**Slave Select:** Active-low select for the onboard slave resources (the shared/dual ported slaves in a master/slave application). Derived from MASN and ONBD, or from ASN and SLVN. If necessary, MASTENN and VMEENN are cycled to provide address set-up time before SLVSELN is asserted.
BRN	23	37	O	M,MS	**Bus Request:** Active-low, open collector VMEbus request. Direct connect to the selected level among VMEbus BR0* – BR3*.
BGINN	27	43	I	M,MS	**Bus Grant In:** Direct connect to the selected level among VMEbus BG0IN* – BG3IN*.

Advance Information

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	CONFIG	NAME AND FUNCTION
	DIP	PLCC			
BGOUTN	26	42	O	M,MS	**Bus Grant Out:** Direct connect to the selected level among VMEbus BG0OUT* – BG3OUT*.
BBSYN	24	38	O	M,MS	**Bus Busy:** Active-low, open collector direct connect to VMEbus BBSY*.
LBGN	4	5	I/O	M,MS	**Local Bus Grant:** Active-low, open collector. Can be connected to the bus grant input of a DMA controller. Asserted when LBRN is low and the BUSCON has control of the VMEbus. Grounded, or driven low during RESET, to prevent the ASN state being driven onto MASN when the BUSCON is not in control of the VMEbus.
RELSE	10	16	I	M,MS	**Release:** Active-high signal indicating that the onboard logic wants to release control of the VMEbus. In DMA controller applications, the BGACKN output of the DMAC should be connected to (or positive-logic ANDed into) this signal.
DTACKN	19	30	I/O O	M,MS S	**Data Transfer Acknowledge:** Active-low, open collector. Direct connect to VMEbus DTACK*.
BERRN	22	35	I/O O	M,MS S	**Bus Error:** Active-low , open collector. Direct connect to VMEbus BERR*.
LDTACKN	7	11	O, I/O I	M, MS S	**Local DTACK:** Act ive-low, open collector. Output to onboard master and/or input from onboard slave.
LBERRN	6	9	O, I/O I	M, MS S	**Local Bus Error:** Onboard active-low, open collector. Output to onboard master and/ or input from onboard slave.
DSI	3	4	I	All	**Data Strobe:** The high-active or of the onboard data strobes, which may be from the onboard master or VMEbus master.
DSENN	5	8	O	M,MS	**Data Strobe Enable:** Low-active, used to enable the onboard data strobes onto the VMEbus.
R/WN	13	21	I	All	**Read/Write:** Onboard R/W signal from the onboard master or VMEbus master.
DDIR	15	24	O	All	**Data Direction Control:** Direction control for VMEbus data transceivers. A high level indicates the "onboard-to-VMEbus" direction.
DENN	9	15	O	All	**Data Enable:** Low-active enable for VMEbus data transceivers.
RESETN	18	27	I	All	**RESET:** Low-active reset. Clears BUSCON logic.
V_{CC}	8	13	I	All	**Power Supply:** +5 volts.
GND	21	33	I	All	**Ground:** 0V reference.

ADDRESS DECODING

Both the VMEbus and current high-speed processors provide short address-to-strobe set-up times, such that with all but the most simple decode schemes, designers must provide for delaying the strobe until decoder outputs have become valid. However, BUSCON operates as a finite-state machine and must synchronize address strobes and other inputs before it can act on them. The BUSCON design allows this synchronization time to be overlapped with address decoding.

In general, most BUSCON inputs do not have critical timing parameters. Exceptions are the three address decode signals. Figure 6 shows a somewhat simplified model of the VMEbus slave selection logic in the SCB68172. The ASN signal is qualified and sampled by flip-flops A and B on each rising edge of CLK. Flip-flop C is set when ASN is high between cycles, and cleared by a falling edge on DTACKN or BERRN.

On the rising edge of CLK after flip-flop B samples ASN low, if C is still set and SLVN is low, flip-flop D is set, indicating slave selection. (In reality, there are more terms in the logic to set D.) Once D is set, it remains set until flip-flop A samples ASN high and a similar circuit (not shown) samples DSI low.

Since SLVN is a direct input to flip-flop D, it must meet a set-up time to the clock after ASN is sampled low. Viewed asynchronously, SLVN should be valid slightly less than one clock period after ASN goes low, through shortly after DTACKN goes low.

The onboard logic driven by MASN, VMEN, and ONBD is similar but not as complex. Neither flip-flop C nor a data-strobe-related signal are used, and there are separate flip-flops corresponding to D for each of the VMEN and ONBD signals. VMEN and ONBD should be valid slightly less than one clock period after MASN goes low, through shortly after MASN goes high.

Because ASN and MASN are used directly to clear the corresponding "flip-flop B", their minimum high times are relatively short. However, for consecutive cycles, the inactive time of "flip-flop D" (and signals derived from it) will be at least two clock periods because of the feedback path from "D" to "B".

VMEbus ARBITRATION

BUSCON begins VMEbus arbitration by driving BRN low if MASN, VMEN and ONBD indicate a VMEbus cycle (or if LBRN goes low) and the BUSCON does not have control of the bus.

After driving BRN, BUSCON waits for the BGINN input which is connected to the selected one among the four VMEbus arbitration levels. (During this time it can of course respond to cycles from other VMEbus masters.) When it receives BGINN low while holding BRN low, it drives BBSYN low and thereafter releases BRN. (If it receives

Advance Information

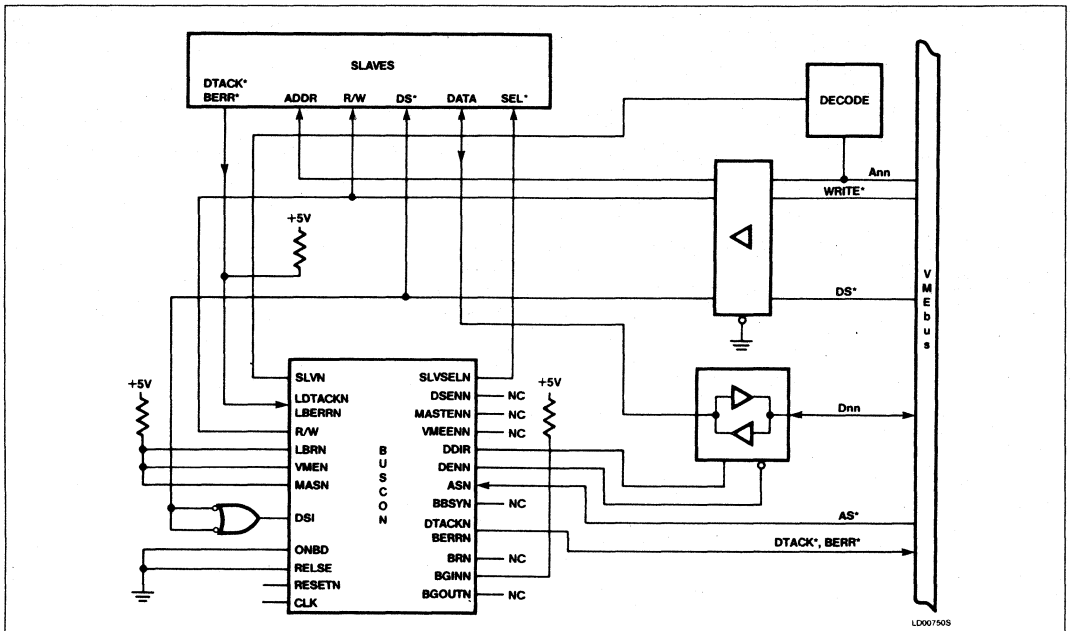


Figure 1. Slave-Only Application

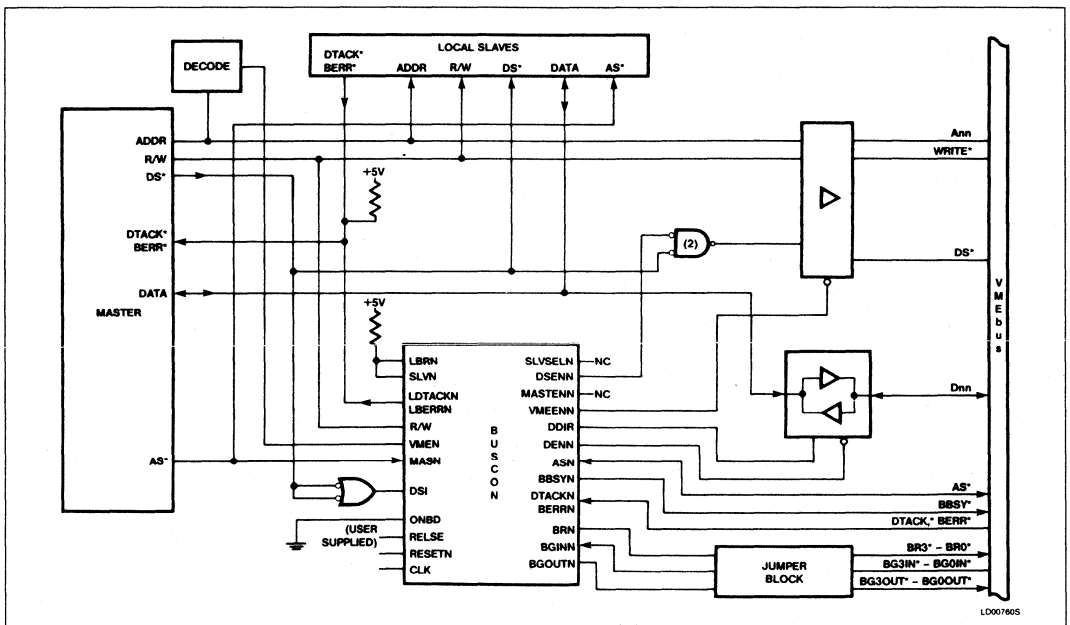


Figure 2. Master-Only Application

Advance Information

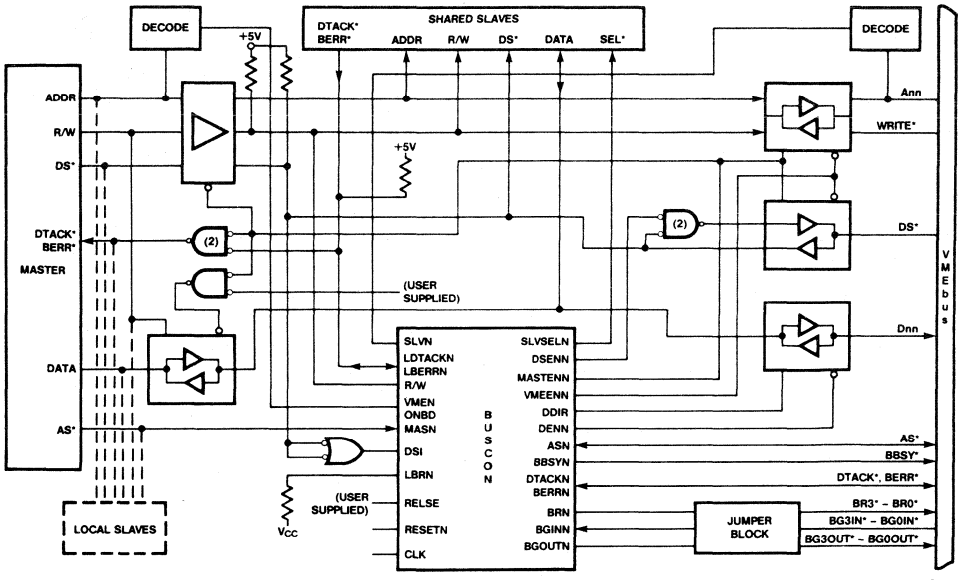


Figure 3. Processor-Type Master/Slave Application

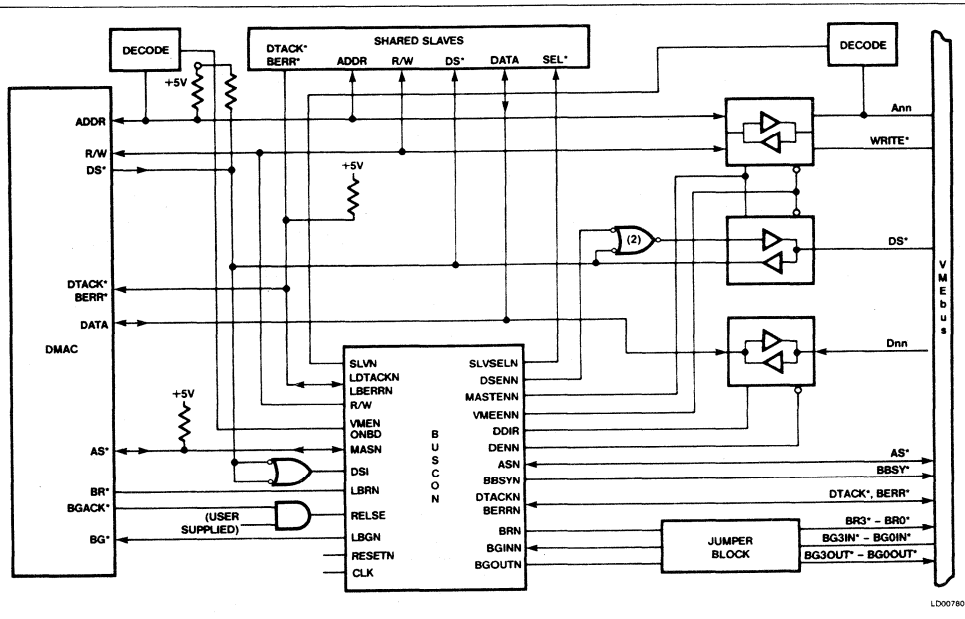


Figure 4. DMAC-Type Master/Slave Application

Advance Information

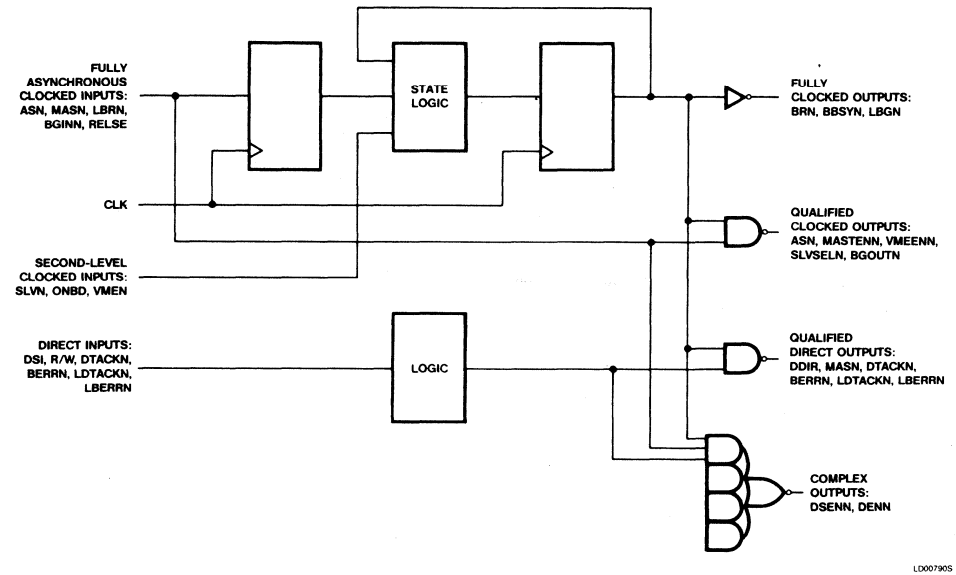


Figure 5. Functional Model

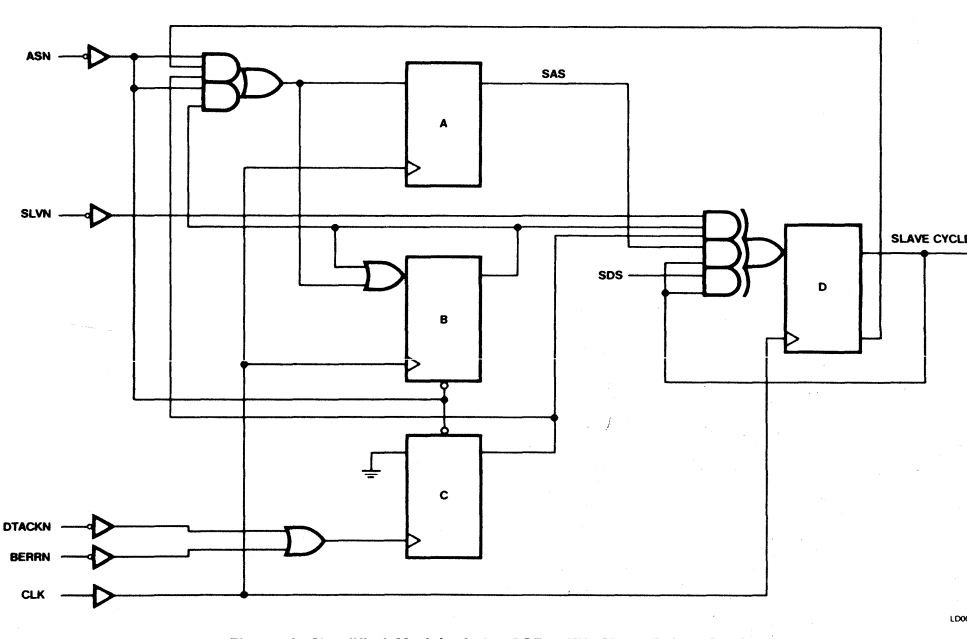


Figure 6. Simplified Model of the SCB68172 Slave Select Logic

Advance Information

BGINN low at any other time, it drives BGOUTN low and continues to do so until BGINN goes high.)

Once BUSCON has driven BBSYN low, it waits for any current VMEbus cycle to complete as evidenced by ASN high. Then it begins to drive ASN (initially high) and drives VMEENN low to enable the address out onto the VMEbus.

If the BRN was initiated by MASN, BUSCON then waits two clock periods for address set-up time before driving ASN low. If the bus acquisition was initiated by LBRN, it waits for MASN.

BUSCON will release the BBSYN signal on a rising clock edge at which all of the following conditions are met:

1. It is at least three clock periods after the edge on which BBSYN was asserted, and
2. Any prior master's cycle has completed and VMEENN has been driven low, and
3. The BGINN input was high on the last rising clock edge, and
4. The RELSE input was high on the last rising clock edge, and
5. It is not the clock edge at which BUSCON asserts ASN, and
6. It is not the clock edge at which BUSCON withdraws ASN, and
7. LBRN was high on the last rising clock edge.

If BBSYN is released while BUSCON is not driving ASN low, then VMEENN goes high when BBSYN is released, to release the VMEbus. Otherwise, VMEENN goes high shortly before ASN goes high.

RELSE is provided to allow user determination of the method of VMEbus release. The BGACKN output of a DMA controller can be connected to (or included in) this signal to allow the device to control how long it keeps the bus. The OR of the VMEbus requests can be connected to (or included in) this signal for release on request (ROR) operation. If RELSE is connected to a constant logic high, BUSCON will release the bus as soon as possible, i.e. during the first bus cycle.

VMEbus MASTER OPERATION

When the BUSCON has VMEbus control, VMEbus cycles indicated on MASN and VMEN produce ASN low on the VMEbus. DDIR and DENN control the VMEbus data transceivers. DDIR reflects the R/WN line.

In a write operation, DENN is driven low to drive data onto the VMEbus whenever the BUSCON has control of the VMEbus, R/WN is low, and the previous VMEbus slave has released DTACKN and BERRN to high. (The DTACKN/BERRN requirement does not apply to subsequent cycles among consecutive writes, if R/WN is maintained continuously low.) DSENN is then driven low when DENN has been low for more than a clock period, and DTACKN and BERRN are high, but not before ASN is driven low. DSENN goes high after DSI goes low or MASN goes high, whichever occurs first. DENN goes high after R/WN goes high, or with VMEENN going high, whichever occurs first.

In a read operation (R/WN is high), DENN goes low to drive data in from the VMEbus after ONBD and VMEN have been sampled, DSI is high, and MASTENN is low. DSENN goes low after DSI, DTACKN, and BERRN are all high, but not before ASN goes low. DSENN and DENN go high after DSI goes low or MASN goes high, whichever occurs first.

DTACKN and BERRN are inputs and drive LDTACKN and LBERRN as outputs. LDTACKN and LBERRN are released when the onboard master makes DSI low. If the VMEbus slave continues to hold DTACKN or BERRN low thereafter, DSENN, LDTACKN and LBERRN are inhibited for the next cycle until the response is released.

MASTENN and VMEENN are kept low while the BUSCON has VMEbus control. The MAS-to-AS delay thus provides automatic address-set-up time for subsequent VMEbus cycles.

The need to transceive the data strobes in a master/slave application, plus qualify the onboard master's strobes with DSENN for output, can be accomplished in several ways as shown in figure 7.

MASTER/SLAVE SWITCHING

BUSCON includes arbitration and switching logic between VMEbus slave cycles and onboard master cycles (to a shared onboard slave or the VMEbus). The logic remains in its previous direction until forced to switch by another cycle. This provides minimum overhead for slave-only or master-only operation, and for consecutive cycles from the same master.

If a master cycle to a shared slave occurs, or BGINN arrives when requesting the VMEbus, after a slave cycle with another VMEbus master, VMEENN goes high to disable the address from the VMEbus. On the next clock edge, MASTENN goes low to enable the master's address back out onto the onboard bus.

For a VMEbus master cycle, if the current VMEbus cycle is also over, VMEENN then goes low to enable the address out onto the VMEbus.

For a master cycle to a shared slave, SLVSELN goes low one clock period after MASTENN goes low, or if the master direction is continuing, after ONBD is sampled high. SLVSELN goes high shortly after MASN goes high. DTACKN and BERRN are isolated from LDTACKN and LBERRN. DSENN is kept high. DENN is kept high except in a write cycle when BUSCON has VMEbus control.

If an onboard master cycle and VMEbus slave cycle both arrive for the shared slaves within the same clock period, the previous direction of the master/slave switch is retained.

VMEbus SLAVE OPERATION

If a VMEbus slave cycle occurs after a master cycle, or while BUSCON is requesting the VMEbus, MASTENN goes high, and on the subsequent clock VMEENN goes low to enable the VMEbus address and control signals onto the board.

SLVSELN goes low one clock period after VMEENN goes low, to signal the shared slave(s) that a cycle is occurring. If the slave mode is continuing, SLVSELN goes low after SLVN is sampled low. SLVSELN goes high shortly after ASN goes high.

DDIR reflects R/WN (in the opposite sense from master operation). LDTACKN and LBERRN are inputs and drive DTACKN and BERRN as outputs.

In write operations, DENN is driven low (to enable data in) whenever R/WN is low and LDTACKN and LBERRN are high. When switching between master and slave operation with R/WN low, DENN sequences like VMEENN.

In read operations, DENN is driven low (to enable data out) after SLVN has been sampled low, and while R/WN and DSI are both high.

SLAVE-ONLY USE

This is the simplest application of the BUSCON. However, handling of board-selection logic from a simple VMEbus address decode, plus driving and sequencing of DTACKN and BERRN, can save VMEbus designers cost and board space even in this application.

SLAVE DESIGN

In the MS and S configurations, slaves operate off the data strobes and SLVSELN rather than address and data strobes. It should be noted that SLVSELN will typically go low after the data strobes go low. Data should not be written nor placed on the data lines until SLVSELN goes low.

Advance Information

68000 DUAL-PORTED OPERATION

BUSCON is ideal for use on a VMEbus board containing a 68000 processor. The obvious approach to dual-porting memory and other slaves, on a board with a 68000, is to use the BRN input of the 68000 to suspend processor operation while another master accesses the onboard slave. This works fine except when the processor has already started a cycle for the VMEbus. This latter coincidence threatens a "deadlock" situation and requires that the processor be "rolled back" off the board's shared bus so that the other master's cycle can occur first. The 68000 has a feature which can be used for this; assertion of both its BERRN and HALTN inputs cause it to suspend operation and retry the cycle when the inputs are released.

The BUSCON does not use these features because there is a flaw in the retry logic. The retry logic does not function during an indivisible RMW sequence (TAS instruction), even in the read cycle. Instead, the assertion of BERRN and HALTN causes an actual bus error exception. It is believed that there is no reliable and general programming solution to the problem of finding the start of the TAS instruction for restart. With 6801x processors, the situation is better because the TAS can be restarted. In any case BUSCON elects to isolate the processor with a few more packages rather than adding complexity to the error-handling software because of dual-ported design.

DMA USE

The BUSCON can be used for VMEbus boards which contain a DMA controller but no processor. Such DMA applications are always master/slave due to the need to program the DMA controller from the VMEbus. There are two operational features of the SCB68172 that are intended for use with DMA controllers. First, the LBRN input can be used to request control of the VMEbus directly, rather than waiting for MASN low and VMEN low as in a processor application. Second, the SCB68172 samples the state of the LBGN pin when RESETN is low. If LBGN is low at the end of RESETN, MASN is used as an input only. If LBGN is high (at the end of the RESETN pulse), the BUSCON thereafter drives the state of VMEbus ASN onto MASN whenever it does not have control of the VMEbus.

For 68000 family DMA controllers, MASN is connected directly to the controller's address strobe pin. The VMEbus ASN-to-MASN feature satisfies the requirement of some DMA controllers that ASN be low on cycles which program them, and also serves to delay the activity of a controller which gets an LBGN response during the last VMEbus cycle by another master.

When LBRN is sampled low, if the BUSCON has retained VMEbus control from previous DMA activity, it continues to retain control for the duration of LBRN being low, and drives LBGN low on the next clock.

Otherwise, it drives VMEbus BRN low on the next clock. When BGINN is sampled low, BBSYN is driven low on the next clock. LBGN is driven low on the same clock as BBSYN if VMEbus ASN is high. If ASN is low, LBGN is

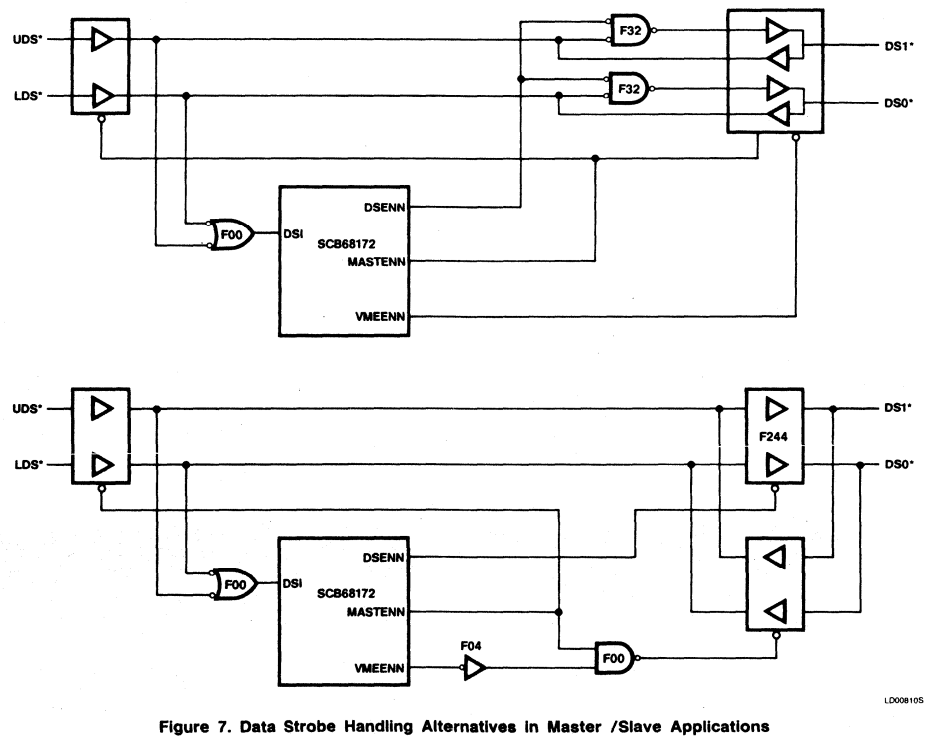


Figure 7. Data Strobe Handling Alternatives in Master /Slave Applications

Advance Information

driven low one clock after BBSYN, except when ASN low and BGINN are both sampled low for the first time in the same clock period and the VMEbus cycle addresses this board – in this last case, LBGN is driven low two clocks after BBSYN. In either of the last two cases, VMEbus ASN low makes MASN low before LBGN goes low, which keeps the DMA controller from starting until the current VMEbus cycle is over.

Note that the local bus request/grant logic and ASN-to-MASN drive feature are separate capabilities, either or both of which can be used in applications not involving a DMA controller. However, note also that when the state of the ASN is driven onto MASN, this is done without conditioning by the state of the master-slave switch. This means MASN can go low before MASTENN and VMEENN have been cycled to bring the VMEbus address onto the board. (The low state of SLVSELN indicates that the VMEbus address is valid on the board.)

DMA applications are always considered master/slave due to the need to program the DMA controller. The BUSCON assumes that it must always have VMEbus control before answering an LBRN with LBGN. If this is not desired, i.e., if the DMA controller will sometimes be programmed to do onboard transfers solely and the designer wishes to optimize for this case, then a processor-type interface should be used, and isolation devices and additional onboard logic are required.

INTERRUPT HANDLING

When a processor handles interrupts from onboard sources and from the VMEbus, the design must include logic to decide whether an interrupt acknowledge cycle is an onboard or offboard cycle. This logic is quite different from the address decoding logic used to make this decision on other cycles.

Performance can be maximized if the interrupt logic can provide ONBD and VMEN within the specified time after MASN goes low, or if the signals can be made to meet their specified set-up and hold times for CLK. In this case ONBD and VMEN need be selected between the IACK and non-IACK sources. Otherwise (i.e., if the interrupt logic presents these signals slowly and asynchronously), MASN must also be selected between the IACK and non-IACK sources.

MASTER BLOCK TRANSFER

The block transfer feature of the VMEbus allows considerable performance improvement for transferring a block of consecutive memory locations. The BUSCON can be used for block transfer operations in the master role.

Master block transfers are applicable to cache subsystems or block transfer on processor boards, and to DMAC-type designs. The only requirements for master block transfers operation with the BUSCON are that external logic must place a block transfer address modifier (AM) code on the VMEbus, and then hold MASN low across a number of data strobes. (Note that a long block transfer can compromise the operation of other VMEbus masters. One strategy to avoid such problems could be to do a minimum of 4 or 8 transfers without interruption, and then switch to release on request (ROR) operation.)

A sample circuit for master block transfers is shown in figure 8. Here, a block transfer is triggered whenever the DMAC accesses a certain range of addresses. The SEQ signal could of course be generated in other ways.

SLAVE BLOCK TRANSFERS

VMEbus slaves that are capable of block transfers latch the bus address into a set of counters on the leading edge of ASN, and then increment the address for each data transfer. The SCB68172 can be used on such slave boards in accordance with revision C of the VMEbus specification.

The revision C specification introduces a limitation on block transfers, namely that a master is not allowed to continue a block transfer across a 256-byte boundary. This limitation has a number of advantages, including reducing the number of counter devices needed on slave boards, allowing straightforward use of page or static column modes on dynamic memories, providing periodic windows in a long block transfer in which higher-priority masters can gain bus control, and (effectively) preventing a block transfer from crossing from one slave board to another.

It is this last advantage that is of particular importance for the SCB68172. It means that VMEbus slaves can make a positive selection-decision after ASN goes low, and this decision will remain valid for the duration of the cycle even if it is a block transfer cycle.

In a block transfer which selects an SCB68172-based slave board, SLVSELN remains low through the block, until ASN goes high. The onboard slave logic then uses the data strobes to define each data transfer.

The data strobe and acknowledge signals are handled in a high- speed combinatorial fashion by the SCB68172 in both the master and slave roles. Block transfers are inherently faster on the VMEbus because the address need be passed and decoded only once, and because the slave can look ahead (pipeline) subsequent transfers in a block read cycle. With the SCB68172, this inherent speed advantage is augmented by the advantage of combinatorial over sequential (arbitrated) logic.

TRI-STATE ENABLE SWITCHING (MASTENN, VMEENN, DENN)

As a result of speed optimization of master/slave switching, some parts used in PMS applications may exhibit short high-going transients on MASTENN, VMEENN, and/or DENN, if requests for access to the shared slave(s) arrive closely in time from both the onboard master and the VMEbus master. These transients should pose no problem as long as the signals are used as intended (i.e., as tristate enables). The following points apply:

1. A transient will occur only when SLVSELN has been high for at least one clock period, and at least one clock period before a subsequent low on SLVSELN.
2. A transient will occur only if the current master/slave direction is maintained.
3. Low-going transients (which could cause tristate conflicts) do not occur.
4. Commonly such transients will be eliminated by external capacitance, and/or rejected by receivers on other parts. In any case the logic levels on signals controlled by these enable signals should not be affected.
5. Edge-sensitive use of these signals is inadvisable in a PMS application.

Advance Information

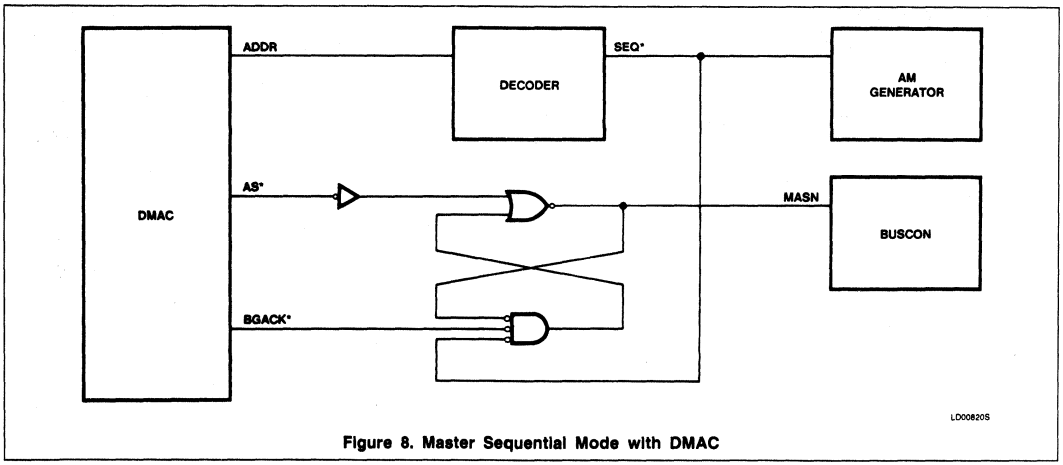
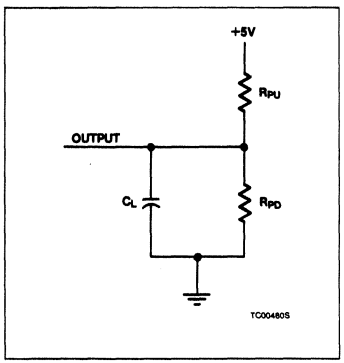


Figure 8. Master Sequential Mode with DMAC

TEST CONDITIONS

Unless otherwise noted, the following timing parameters are based on loading as follows:



SIGNALS	R_{PU}	R_{PD}	C_L
ASN, BRN, BBSYN, DTACKN, BERRN LDTACKN, LBERRN, LBGN, BGOUTN, VMEENN, SLVSELN, DSENN, DENN,	150	235	300
DDIR	2K	N/A	15
MASTENN	280	N/A	45
MASN	180	1K	45

Advance Information

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Supply voltage	−0.5 to +7.0	V
Input voltage	−0.5 to +5.5	V
Operating temperature range[2]	0 to +70	°C
Storage temperature	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V±5%, T_A = 0°C to +70°C[3,4]

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
DSI, ONBD, SLVN, VMEN, LBRN, RELSE, RESETN					
I_{IL}	Input low current	V_{IN} = 0.4V		−400	μA
I_{IH}	Input high current	V_{IN} = 2.7V		20	μA
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
ASN, MASN, BGINN, DTACKN, BERRN, LDTACKN, LBERRN					
I_{IL}	Input low current	V_{IN} = 0.4V		−400	μA
I_{IH}	Input high current	V_{IN} = 2.7V		20	μA
V_{TH+}	High-going threshold voltage		1	1.65	V
V_{TH-}	Low-going threshold voltage		0.8	1.15	V
$V_{TH+} - V_{TH-}$	Hysteresis		0.2		V
R/WN, CLK					
I_{IL}	Input low current	V_{IN} = 0.4V		−800	μA
I_{IH}	Input high current	V_{IN} = 2.7V		40	μA
V_{IL}	Input low voltage			0.8	V
V_{IH}	Input high voltage		2.0		V
BGOUTN, VMEENN, SLVSELN, DSENN, DENN, DDIR (low current totem pole)					
V_{OL}	Output low voltage	I_{OL} = 8mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −0.4mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−15	−100	mA
MASTENN (high current totem pole)					
V_{OL}	Output low voltage	I_{OL} = 24mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −2.6mA, V_{CC} = 4.75V	2.4		V
V_{OH}	Output high voltage	I_{OH} = −1mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−40	−100	mA
MASN (low current tristate)					
V_{OL}	Output low voltage	I_{OL} = 8mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −0.4mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−15	−100	mA
I_{OZL}	Three-state-off leakage current, low level	V = 0.4V		−21	μA
I_{OZH}	Three-state-off leakage current, high level	V = 2.7V		20	μA
ASN (high current tristate)					
V_{OL}	Output low voltage	I_{OL} = 64mA, V_{CC} = 4.75V		0.5	V
V_{OH}	Output high voltage	I_{OH} = −7.8mA, V_{CC} = 4.75V	2.4		V
V_{OH}	Output high voltage	I_{OH} = −3mA, V_{CC} = 4.75V	2.7		V
I_{OS}	Short-circuit output current	V_{OUT} = 0V	−120	−300	mA
I_{OZL}	Three-state-off leakage current, low level	V = 0.4V		−60	μA
I_{OZH}	Three-state-off leakage current, high level	V = 2.7V		60	μA

Advance Information

DC ELECTRICAL CHARACTERISTICS (Continued)

PARAMETER		TEST CONDITIONS	LIMITS		UNIT
			Min	Max	
LDTACKN, LBERRN, LBGN (low current open collector)					
V_{OL}	Output low voltage	I_{OL} = 8mA, V_{CC} = 4.75V		0.5	V
I_{OH}	Output high current	V_{OUT} = 5.5V		100	µA
DTACKN, BERRN, BRN, BBSYN (high current open collector)					
V_{OL}	Output low voltage	I_{OL} = 40mA, V_{CC} = min		0.4	V
V_{OL}	Output low voltage	I_{OL} = 70mA, V_{CC} = min		0.5	V
I_{OH}	Output high current	V_{OUT} = 2.7V		60	µA
I_{OH}	Output high current	V_{OUT} = 5.5V		250	µA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other conditions other than those indicated in the Electrical Characteristics section of this data sheet is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over specified temperature range.
4. All voltage measurements are referenced to ground (V_{SS}). For testing, all signals swing between 0.4V and 2.4V with a transition time of 10ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V as appropriate.

AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V±5%, T_A = 0°C to +70°C[3,4]

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
Clock and general parameters						
1	9-18, 20	CLK cycle time (clk)	TBD		ns	
2	9-18, 20	CLK low time	TBD		ns	
3	9-18, 20	CLK high time	TBD		ns	
Asynchronous input set-up time to CLK high						
4	9, 10, 11, 12, 13, 18, 20	ASN, MASN low	15		ns	5
5	9, 10, 11, 13	ASN, MASN high	14		ns	5
6	9, 10, 11, 12, 13, 20	SLVN, VMEN low	22		ns	6
7	11, 17, 18	SLVN, VMEN high	20		ns	6
8	9, 10, 13	ONBD low	22		ns	6
9	18	ONBD high	25		ns	6
10	13, 16, 20	LBRN, RELSE, BGINN low	9		ns	5
11	14, 15	LBRN, RELSE, BGINN high	11		ns	5
12	11, 12, 13	DSI low (end of slave cycle)	16		ns	5
Asynchronous input hold time from CLK high						
13		ASN, MASN, DSI	0		ns	7
14		ONBD, VMEN	0		ns	8
15		LBRN, RELSE, BGINN	2		ns	7
Prop, CLK high to:						
16	16	BGOUTN low	12	27	ns	
17	20	LBGN low	12	30	ns	
18		LBGN high	16	41	ns	
19	13, 16, 20	BBSYN, BRN low	17	37	ns	
20	13, 20	BBSYN, BRN high (C_L = 50)	20	47	ns	
		(C_L = 300)	40	68	ns	
21	9, 10, 13	ASN low	15	37	ns	
22	9, 10	ASN high	13	31	ns	
23	11, 12, 13, 18, 20	SLVSELN, VMEENN low	14	35	ns	
24	13, 18, 20	MASTENN low	15	38	ns	
Miscellaneous						
25		RESETN width low	6clk		ns	9
26	16	BGINN low to BGOUTN low	clk+10	2clk+35	ns	
27	16	BGINN high to BGOUTN high	3	10	ns	
28	9, 11	R/WN high to DSI high (start of read cycle)	10		ns	
29	9, 12	DSI low to RWN low (end of read cycle)	10		ns	

Advance Information

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
Address decoding						
30	11, 12, 13, 20	ASN low to SLVN valid		clk–5	ns	
31	9, 10, 13, 18	MASN low to VMEN valid		clk–5	ns	
32	9, 10, 13, 18	MASN low to ONBD valid		clk–7	ns	
33	17	SLVN high after DTACKN low	14		ns	
34	9, 10	VMEN, ONBD valid after MASN high	6		ns	
35	9, 10	MASN high	TBD			10
36	9, 10, 11	DSI low	TBD			
37	11	ASN high	TBD			10
VMEbus acquisition						
38	13	MASN low to BRN low	clk+15	2clk+45	ns	11
38A	13, 20	ASN low to BGINN low (early release by other master)	10		ns	
39	13, 20	BGINN low to BBSYN low	clk+15	2clk+45	ns	
40	13, 20	BBSYN low to BRN high	clk	clk+50	ns	
41	13	BGINN low to VMEENN low, DENN low (write)	clk+12	2clk+40	ns	12
42	13	ASN high to VMEENN low, DENN low (write)	11	31	ns	12,13
43	13	VMEENN low to ASN low	2clk–10	2clk+15	ns	14
43A	13, 16, 20	BBSYN or BGOUTN low to BGINN high	10		ns	
VMEbus master cycles						
44	9, 10	ASN high (successive VMEbus master cycles)	2clk–15		ns	14
45	9, 10	MASN low to ASN low (subsequent cycle retaining VMEbus control)	clk+13	2clk+45	ns	14
46	9, 10	ASN low to DSEN low	–4	5	ns	17, 18
47	10, 13	R/WN low to DDIR high	6	15	ns	
48	10, 13	DDIR high to DENN low (write)	2	7	ns	15
49	10	DTACKN and BERRN high to DENN low (write, 1st bus cycle or preceded by read)	7	21	ns	15
50	10	DENN low to DSENN low (write)	clk+10	2clk+40	ns	17
51	9, 11, 13	R/WN high to DDIR low	6	16	ns	
52	9	DDIR low to DENN low (read)	5		ns	16
53	9	DSI high to DENN low (read)	8	18	ns	16
54	9	DSI high to DSENN low (read)	8	18	ns	18
55	9, 10	DTACKN and BERRN high to DSENN low	6	19	ns	17, 18
56	9, 10	DTACKN or BERRN low to LDTACKN or LBERRN low	6	17	ns	
57	9, 10	LDTACKN or LBERRN low to DSI low or MASN high	TBD			
58	9, 10	DSI low to DSENN high	7	16	ns	19
59	9, 10	MASN high to DSENN high	13	28	ns	
60	10	R/WN high to DSENN high (after a write)	5	14	ns	20
61	9, 10	MASN high to ASN high (unless early release)	clk+11	2clk+38	ns	
62	9	MASN high to DENN high (read)	13	28	ns	21
63	9	DSI low to DENN high (read)	7	16	ns	21
64	10	R/WN high to DENN high (write)	5	14	ns	22
65	9, 10	DSENN high to LDTACKN and LBERRN high	7	24	ns	23
66	9, 10	DTACKN and BERRN high to LDTACKN and LBERRN high	7	23	ns	23
VMEbus release						
67	14	BBSYN low	2clk		ns	
68	14, 15	BGINN high to BBSYN high	clk+18	2clk+70	ns	24
69	14, 15	RELSE high to BBSYN high	clk+20	2clk+72	ns	24
70	14	ASN low to BBSYN high (early release)	clk–25		ns	
71	14	MASN high to VMEENN high (early release)	8	20	ns	
72	14	MASN high to DENN high (early release, write)	8	20	ns	
73	14	DENN (write) and VMEENN high to ASN high (early release)	5	15	ns	
74	14	ASN high to ASN released (early release)	5	20	ns	
75	15	ASN high to BBSYN high (intercycle release)	clk–10		ns	
76	15	DENN (write) and VMEENN high to BBSYN high (intercycle release)	5	30	ns	
77	14, 15	BBSYN high to RELSE low	0		ns	

Advance Information

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
Master to slave switching						
78	11, 13, 20	(External) ASN low to MASTENN high	10	28	ns	25
79	11, 13, 20	SLVN Low to MASTENN high	7	17	ns	25
80	11	SLVSELN high to MASTENN high	7		ns	25
81	11, 20	MASTENN high to VMEENN low	14	clk+36	ns	
81A	11, 13	VMEENN low to DDIR change	−5	+5	ns	
82	11, 13, 20	VMEENN low to SLVSELN low	clk−11	clk−5	ns	26
VMEbus slave cycles						
83	12	SLVSELN high (successive slave cycles)	2clk−5		ns	26
84	12	ASN low to SLVSELN low (already in slave state)	clk+12	2clk+40	ns	26
85	12, 13	R/WN low to DDIR low	6	14	ns	
86	12, 13	DDIR low to DENN low (write)	5	12	ns	27
87	12	LDTACKN and LBERRN high to DENN low (write)	7	20	ns	27
88	12, 13	R/WN high to DDIR high	5	14	ns	
89	11	DDIR high to DENN low (read)	2	7	ns	28
90	11	ASN low to DENN low (read)	clk+20	2clk+47	ns	28
91	11	DSI high to DENN low (read)	6	16	ns	28
92	11, 12, 13, 18	SLVSELN low and DSI high to LDTACKN or LBERRN low	0		ns	29
93	11, 12, 13, 18	LDTACKN or LBERRN low to DTACKN or BERRN low	10	20	ns	
94	12, 13	LDTACKN or LBERRN low to DENN high (write)	9	22	ns	
94A	11, 12, 13, 18	DTACKN or BERRN low to DSI low or ASN high	0		ns	
95	11	DSI low to DENN high (read)	7	16	ns	
96	11, 12, 18, 20	ASN high to SLVSELN high	13	28	ns	
97	11, 12, 13	DSI low to DTACKN and BERRN high				
		(C_L = 50)	17	37	ns	
		(C_L = 300)	37	60	ns	
98	11, 12, 13, 20	DSI low to LDTACKN and LBERRN high	0	35	ns	30, 32
99	11, 12	LDTACKN and LBERRN high to (next) DSI high	0		ns	30
Slave to master switching						
100	18	MASN low to VMEENN, DENN (VMEbus slave write) high	18	clk+47	ns	31
101	18	ONBD high to VMEENN, DENN (VMEbus slave write) high	8	24	ns	31
102		VMEN low to VMEENN, DENN (VMEbus slave write) high	18	21	ns	31
103	13, 18, 20	SLVSELN high to VMEENN, DENN (VMEbus slave write) high	0		ns	31
104	13, 20	DSI low (selected) to VMEENN, DENN (VMEbus slave write) high	14	clk+40	ns	31
105	13, 18, 20	VMEENN high to MASTENN low	24	clk+20	ns	
107	13, 20	MASTENN low to VMEENN low, DENN low (write, if next cycle on VMEbus)	3		ns	12,15
108	18	MASTENN low to SLVSELN low (if next cycle on board)	clk−13	clk	ns	33
Onboard cycles						
109	18	SLVSELN high (successive onboard cycles)	3clk+4		ns	33
110	18	MASN low to SLVSELN low (MASTENN already low)	clk+17	2clk+42	ns	33
111	18	MASN high to SLVSELN high	12	26	ns	

Advance Information

AC ELECTRICAL CHARACTERISTICS (Continued)

NO.	FIGURE	CHARACTERISTIC	TENTATIVE LIMITS		UNIT	NOTES
			Min	Max		
DMAC-Type operation						
112	19	BBSYN high to MASN active (C_L = 50)	0	5	ns	34
		(C_L = 300)	−25	−15	ns	34
113	19	ASN high to MASN active	13	40	ns	34
114	19	ASN low to MASN low	10	25	ns	
115	19	ASN high to MASN high	6	16	ns	
116	19	ASN high to MASN released	12	32	ns	35
117	19	LBGN low to MASN released	5	12	ns	35
118	20	LBRN low to BRN low (if BBSYN released)	clk+15	2clk+45	ns	
119		LBRN low to LBGN low	clk+10	2clk+37	ns	36
120	20	BGINN low to LBGN low (ASN high)	clk+15	2clk+45	ns	
		(ASN low)	2clk+15	3clk+45	ns	
121	20	MASN low (output) to LBGN low	2clk+12		ns	
122	20	LBGN low to LBRN high	0		ns	
123	20	LBRN high to LBGN high	clk+14	2clk+48	ns	37
124	14, 15	LBRN high to BBSYN high (C_L = 50)	clk+18	2clk+55	ns	24
		(C_L = 300)	clk+35	2clk+75	ns	24
125	20	ASN high to LBGN high (selected)	2clk+18	3clk+50	ns	37
126	20	DSI low to LBGN high (selected)	2clk+20	3clk+52	ns	37

NOTES:

5. These set-up times guarantee recognition at a rising edge of CLK, but the device will operate correctly if they are not met. If the asynchronous input is changed between the set-up and hold times, the new state of the input may be recognized at this clock or the following clock.
6. These set-up times are required on the rising edge of CLK following the one on which ASN or MASN is first recognized low. If parameters 30, 31, and 32 are met, these parameters are automatically guaranteed.
7. These hold times guarantee (continued) recognition of the signal state at a rising edge of CLK, but the device will operate correctly if they are not met.
8. These hold times are required on the rising edge of CLK preceding the one on which MASN is first recognized high. Parameter 34 provides a more straightforward requirement which guarantees these times.
9. This parameter applies after V_{CC} and the clock signal are both within the specified limits.
10. These minimum times are to guarantee recognition. Operation will be limited by 44, 83, and 109 if the strobe is high for less than 2clk.
11. BRN is driven low, and this acquisition sequence applies, only if BBSYN is high.
12. VMEENN is driven low only when 41, 42, and 107 have been met.
13. Applies to ASN of VMEbus cycle which does not select this board as a slave.
14. ASN goes low when 43, 44, and 45 are met. 44 is not applicable on the first cycle after acquiring the VMEbus.
15. In a write operation, DENN goes low when 41, 42, 48, 49, and 107 are met. 49 does not apply for subsequent cycles in a series of writes if R/WN is held low throughout.
16. In a read operation, DENN goes low when 52 and 53 are met.
17. In a write operation, DSENN goes low when 46, 50, and 55 are met. 55 is significant only for subsequent cycles in a series of writes with R/WN held low throughout.
18. In a read operation, DSENN goes low when 46, 54, and 55 are met.
19. DSENN goes high when either 58 or 59 is met.
20. Applies only if R/WN remains low after a write cycle, so that DSENN goes low again.
21. In a read operation, DENN goes high when either 63 or 64 is met.
22. In a write operation, DENN goes high when either 64 or 71 is met.
23. LDTACKN and LBERRN go high when either 65 or 66 is met.
24. BBSYN is always released in response to BGINN, RELSE, and LBRN all high. However, if this condition is detected during a clock period in which the decision to change ASN is made, the release of BBSYN is delayed one clock period so that 70 or 75 is met.
25. MASTENN goes high when 78, 79, and 80 are all met.
26. SLVSELN goes low when 82, 83, and 84 (as applicable) are met.
27. In a write operation, DENN goes low when 86 and 87 are met.
28. In a read operation, DENN goes low when 89, 90, and 91 are met.
29. The onboard slave(s) should wait for both SLVSELN and DSI before driving a response.
30. Since BUSCON itself terminates DTACKN and BERRN when DSI goes low, the onboard slaves must meet this requirement to assure that a "lingering" LDTACKN or LBERRN is not presented as DTACKN or BERRN when the next DSI occurs. 99 is the real requirement – 98 max is derived from it, plus the 40nsec minimum high time of VMEbus data strobes and an allowance for receiver skew.
31. VMEENN goes high only when 100, (101 or 102), 103, and 104 are met. 104 applies only if a VMEbus slave cycle (with this board) is ending.
32. The onboard slave(s) must meet this requirement so that the local response is not inadvertantly presented to the onboard master.
33. SLVSELN goes low when 108, 109, and 110 (as applicable) are met.
34. MASN is driven out of Hi-Z state only when 112 and 113 are met.
35. MASN is released to Hi-Z state only when 116 and 117 are met.
36. The max figure applies only if BUSCON has kept VMEbus control (i.e., if BBSYN is low).
37. LBGN goes high only when 123, 125, and 126 are met, but 125 and 126 apply only if a VMEbus slave cycle (with this board) is in progress.

Advance Information

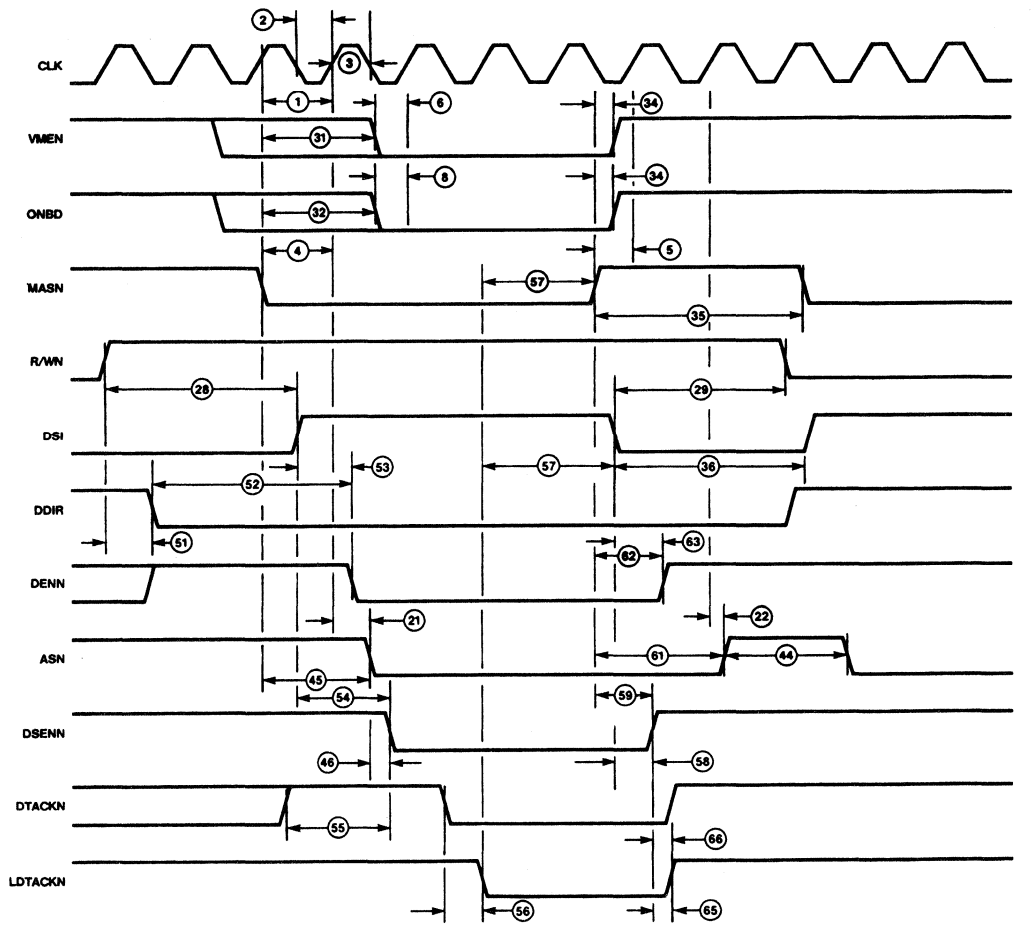


Figure 9. Read Cycle From Onboard Master to VMEbus Slave (Already in Control of VMEbus)

Advance Information

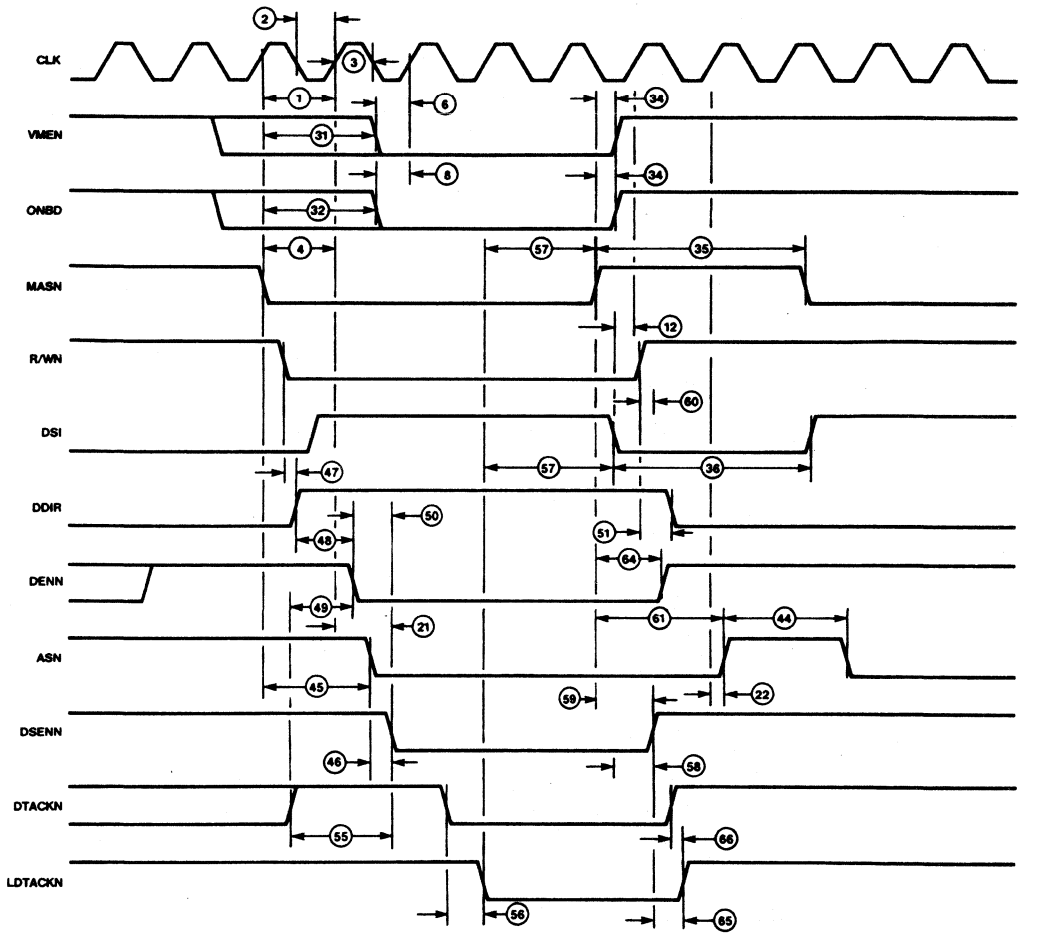


Figure 10. Write Cycle From Onboard Master to VMEbus Slave (Already in Control of VMEbus)

Advance Information

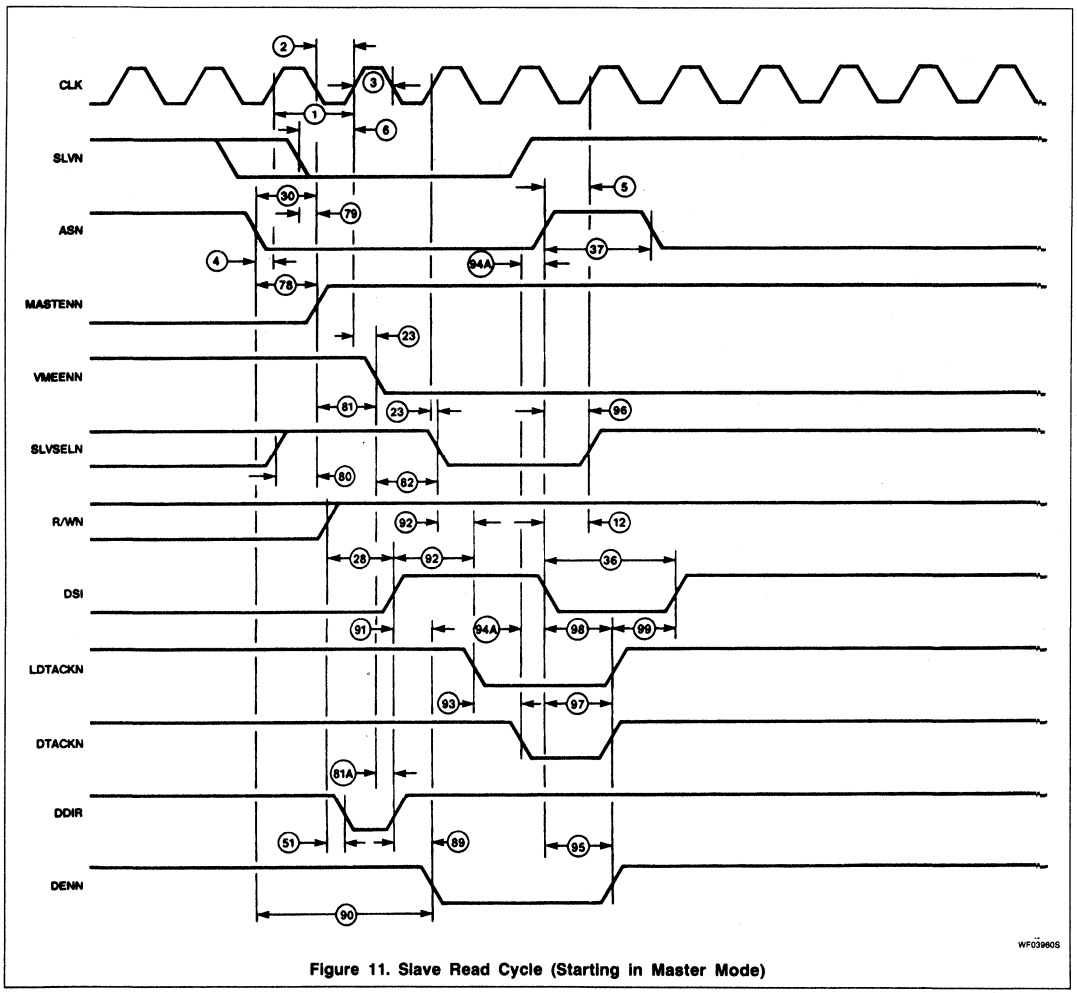


Figure 11. Slave Read Cycle (Starting in Master Mode)

Advance Information

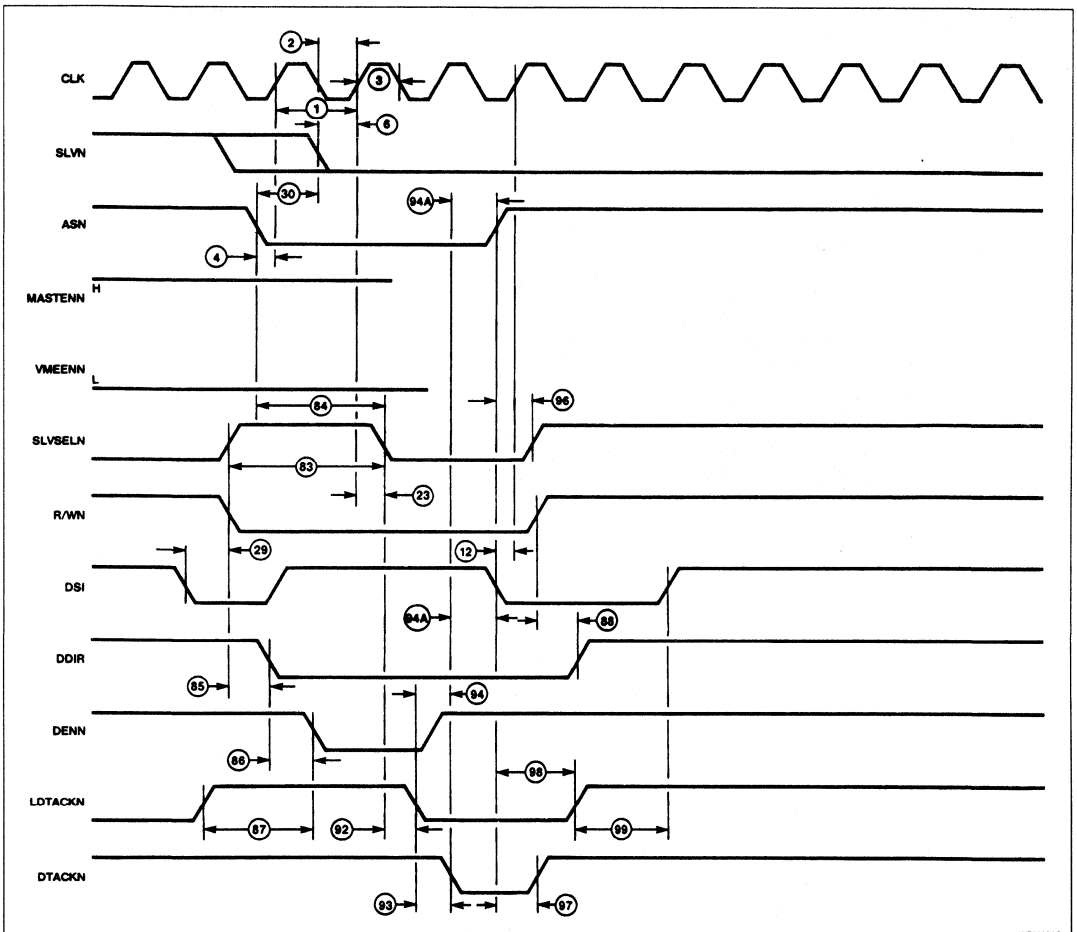


Figure 12. Slave Write Cycle (Starting in Slave Mode)

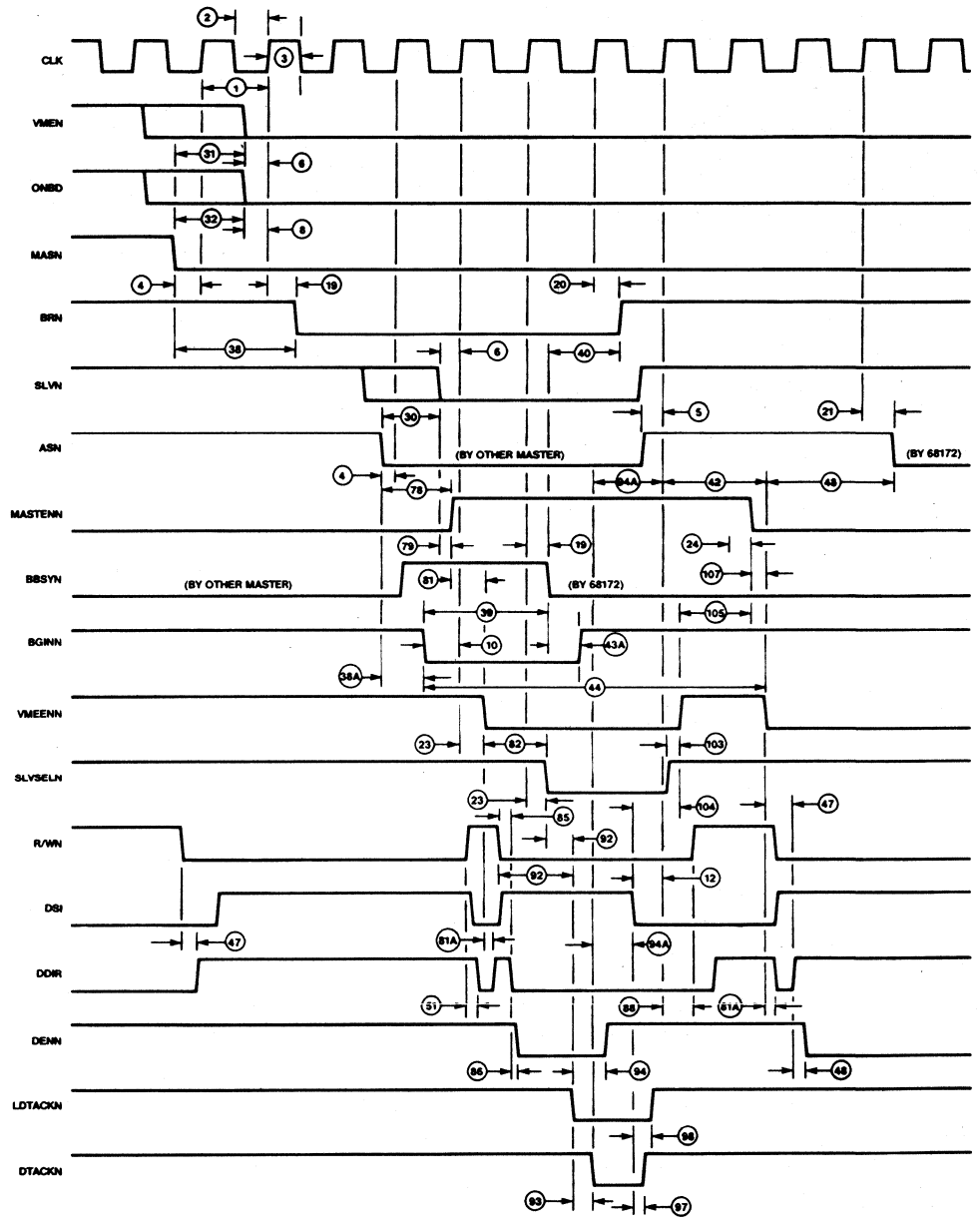


Figure 13. VMEbus Acquisition, Including Slave Write Cycle

Advance Information

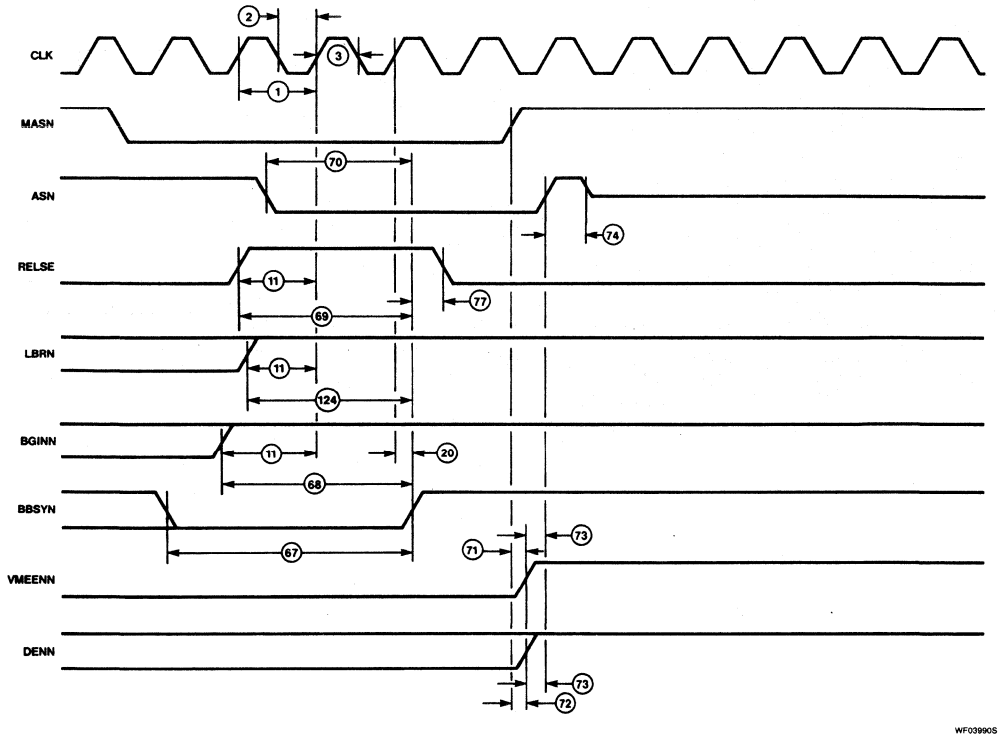


Figure 14. "Early" VMEbus Release

Advance Information

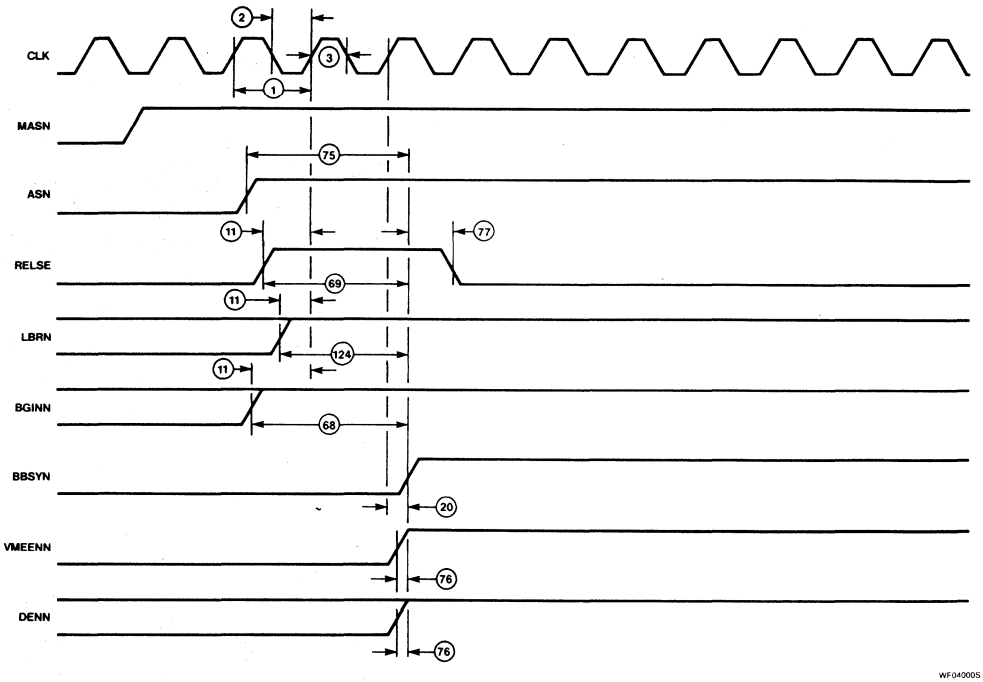


Figure 15. Intercycle VMEbus Release

Advance Information

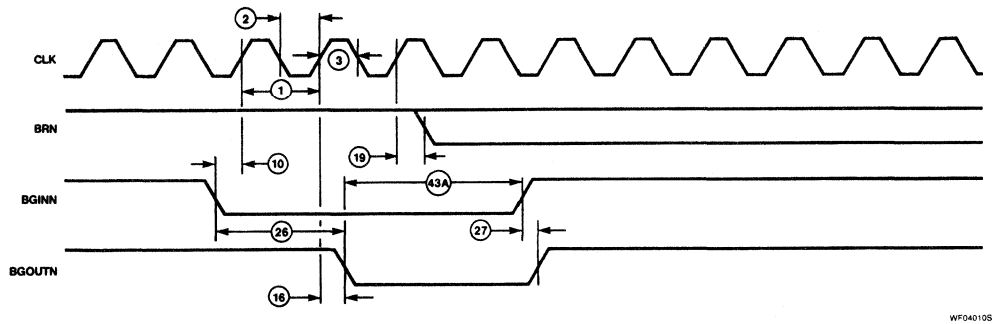


Figure 16. Passing a Bus Grant, Including Earliest Possible Bus Request Thereafter

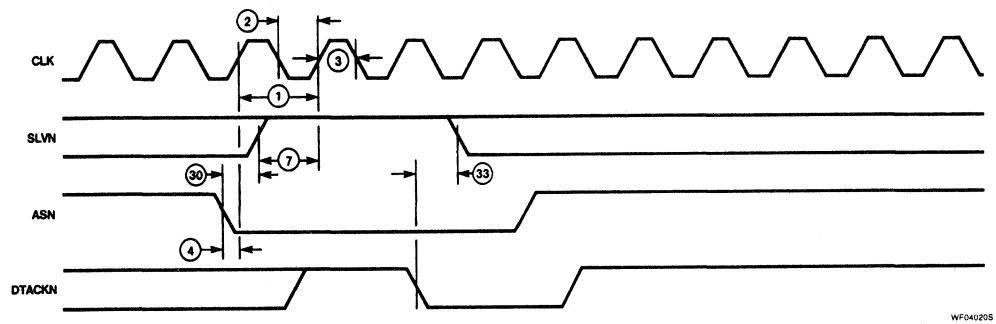


Figure 17. Non-selecting Cycle on VMEbus

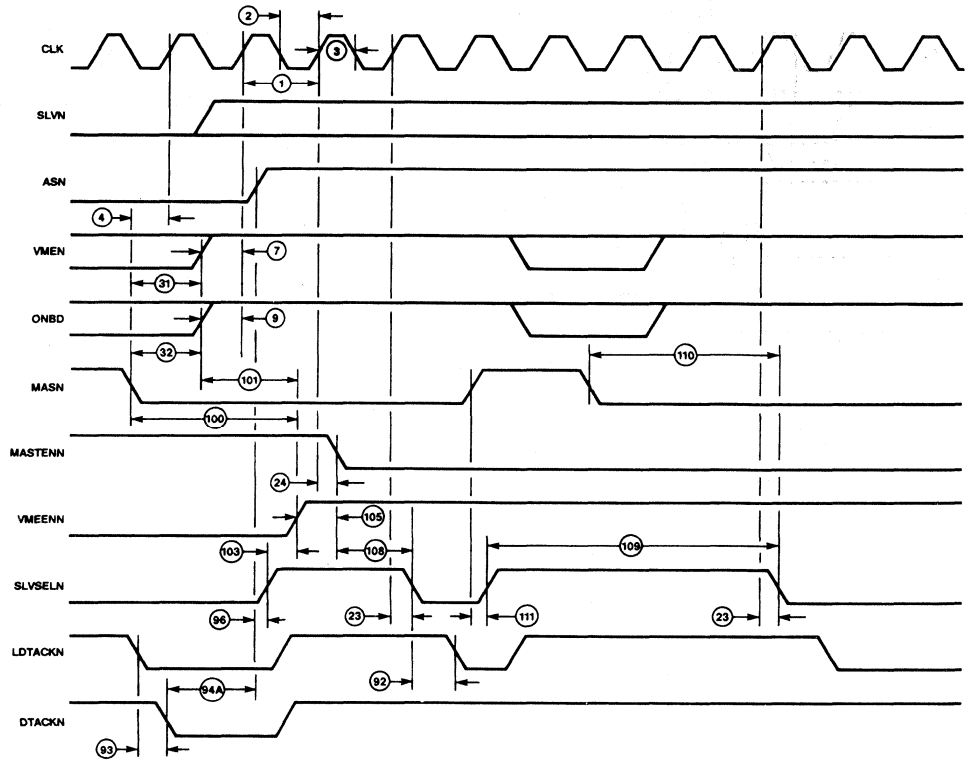


Figure 18. Two Onboard Cycles Following a Slave Cycle

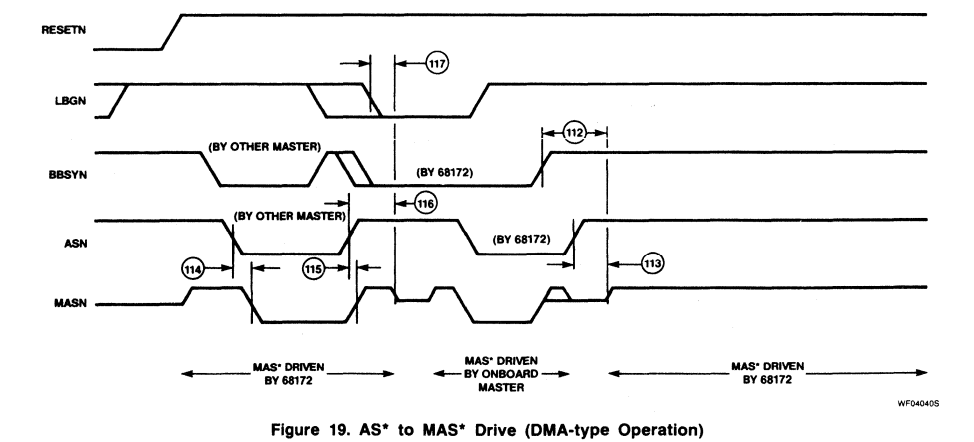


Figure 19. AS* to MAS* Drive (DMA-type Operation)

Advance Information

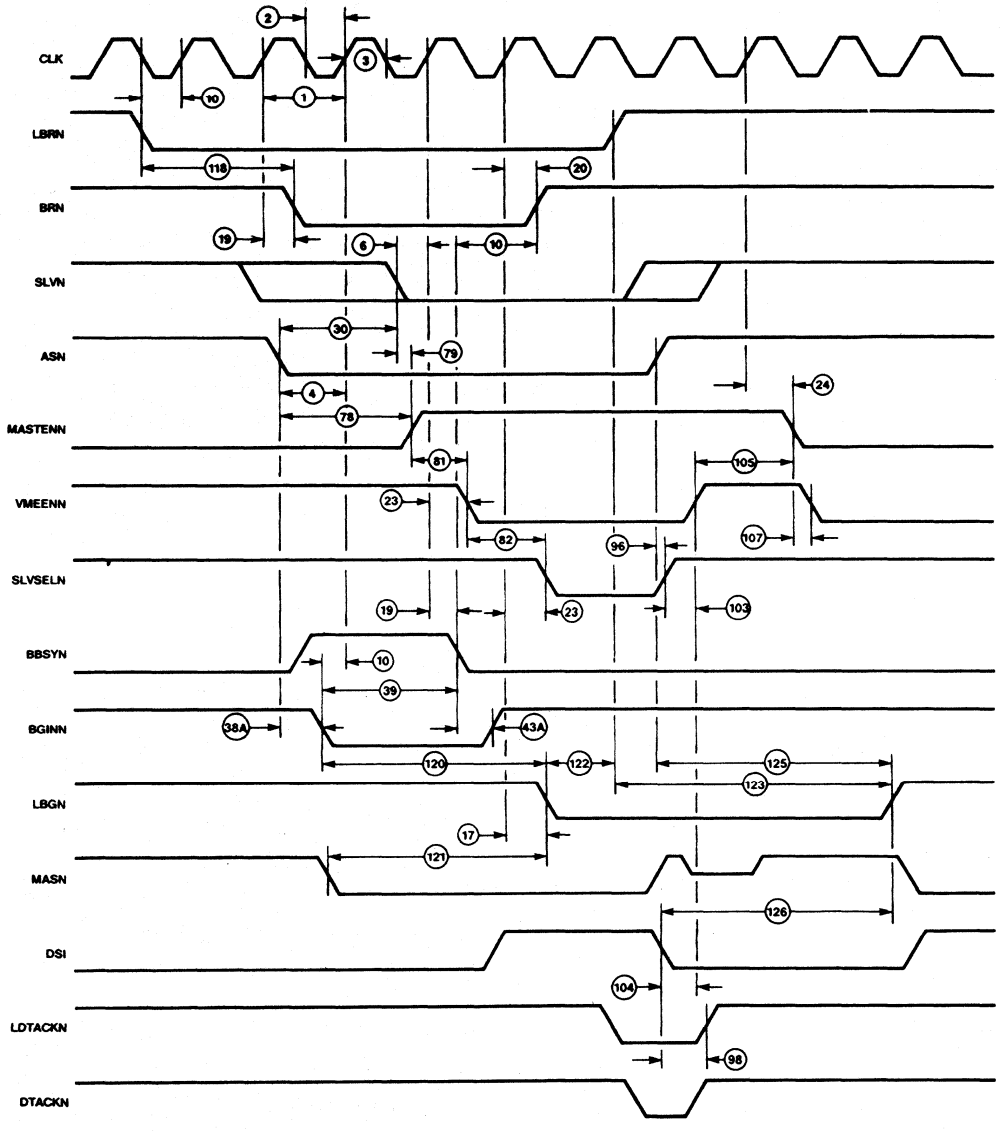


Figure 20. DMA-type VMEbus Acquisition, With Slave Cycle From Another VMEbus Master

DESIGN ENTRY

Bus controller chip lets processor board switch master and slave roles

Carrying the logic needed for a VMEbus interface, this chip eases the design of master-slave boards, with DMA controllers a prime beneficiary.

The development of multiprocessor systems for the VMEbus has brought new design challenges—most notably, the need to coordinate the system's data-handling operations as simply as possible. The task is not always simple, and the bus control circuit usually needs more than a few chips. Accordingly, board space limitations have restricted 90% of all VMEbus-compatible designs to being either master-only or slave-only subsystems.

Boards that can alternate the roles of bus master and bus slave are far more versatile, and the arrival of the first one-chip bus controller makes such master-slave designs far more feasible. The chip incorporates much of the protocol logic required to serve any VMEbus application, whether in processing data, words, and images; in industrial automation and robotics; or in hardware and software development.

Even in non-VMEbus applications, the SCB68172 bus controller (BUSCON) proves valuable when two independent buses must share memory or peripherals. Indeed, viewed generically, it operates as a fairly generalized arbiter and switch between two buses for control of a third bus.

The chip does the work of up to 20 ICs that would normally be needed to create a VMEbus-compatible master-slave processor board. One of the chip's most obvious applications is in classic DMA control—a function often ill-served in existing VMEbus designs by I/O processor architecture. A DMA controller serves as the VMEbus master in normal operation yet must act as the VMEbus slave when programmed by an offboard processor.

Other attractive applications of the chip's master-slave capabilities include its use on a processor board so that on-board memory can be shared (dual-ported) and the processor can accept "write-into" interrupts.

Dual ports make it straightforward to load programs into a processor's on-board memory from elsewhere in the system. They also reduce bus traffic in interprocessor communications schemes—like those that log data into preassigned memory locations, or mailboxes. Finally, dual ports aid the detection and troubleshooting of problems in multiprocessor systems, since a crashed master processor can be used as a slave for diagnostic purposes.

Write-into interrupts are possible only with

Craig MacKenna, Signetics Corp.
Craig MacKenna, now a senior architect with the MOS Microprocessor Division of Signetics in Sunnyvale, Calif., has worked in the computer field for 20 years, gaining varied experience at companies like Astronautics Corp. of America and Mostek. While at Mostek, he was part of the multicompany team that initially designed and set specifications for the VMEbus. In addition, he was the principal author of the VMSbus (VME serial bus) specification.

Reprinted from **ELECTRONIC DESIGN- June 28, 1984**

master-slave processor boards, such as those equipped with the bus controller chips. Used with dual-ported memory, the interrupts can vastly reduce bus traffic. In essence, another VMEbus master can write an interrupt level and vector number into a master-slave processor board in one step. Otherwise, the second master would have to request an interrupt first and then supply the vector information only when the interrupt is acknowledged by the processor.

Other roles

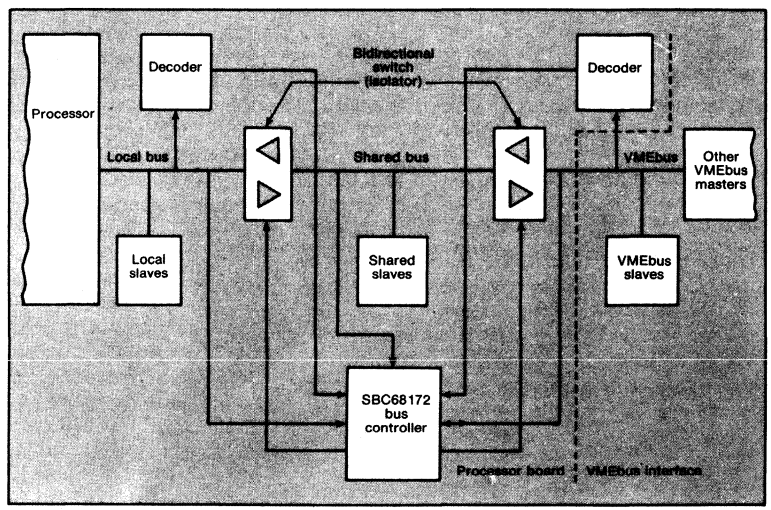
Since the chip has both master and slave capabilities, it can serve equally well in master-only and slave-only designs. The savings in chip count, cost, and board space will be less in those smaller-scale applications, of course. Nonetheless, the device still offers considerable advantages, even in slave-only configurations. For instance, consider the case of a relatively slow address decoder during a short cycle between a fast VMEbus slave and a fast master, where the cycle ends and the address is changed before the slave can decide if it has been selected. The 68172 can reliably decode addresses during such speedy data transfers with other VMEbus masters and slaves.

Moreover, the controller can work with both master and slave sequential operations, which allow high-speed, pipelined accesses to successive memory locations. During such block transfers, it operates purely combinatorially and adds little or no timing overhead.

The bus controller is a bipolar device housed in a 28-pin DIP. It contains no programmable registers and thus is transparent to system software. Two versions are available: one is intended for processor and other general-purpose applications, the other for use with DMA controllers.

In a basic processor master-slave board, the chip directly interfaces a number of VMEbus and processor control signals and also supplies driver and transceiver controls both for other control signals and for addresses and data (Fig. 1). Three buses are involved: the VMEbus, the processor's local bus, and a shared bus.

In operation, the controller sequences its board among three states: In the most typical one, the local bus is connected to the shared bus, which hosts dual-ported, or shared, slaves (Fig. 2a). Consequently, the processor can access resources on its local bus or on the shared



1. In general master-slave designs, the SCB68172 bus controller links shared slaves both with the main processor over a local bus and with other VMEbus masters over shared buses. Switching is governed by a three-state protocol.

bus. On the other hand, the current VMEbus master can access other VMEbus resources but not the shared resources.

In the slave state, the controller connects the VMEbus to the shared bus, which in turn is isolated from the local bus (Fig. 2b). An external VMEbus master can access the board's shared resources, but the processor can access only the resources on its local bus.

When the local bus is connected to the shared bus, which in turn is connected to the VMEbus, the board is in its third state—the current VMEbus master (Fig. 2c). It can now use resources existing on any of the three buses; other VMEbus masters, on the other hand, can access only their on-board slaves.

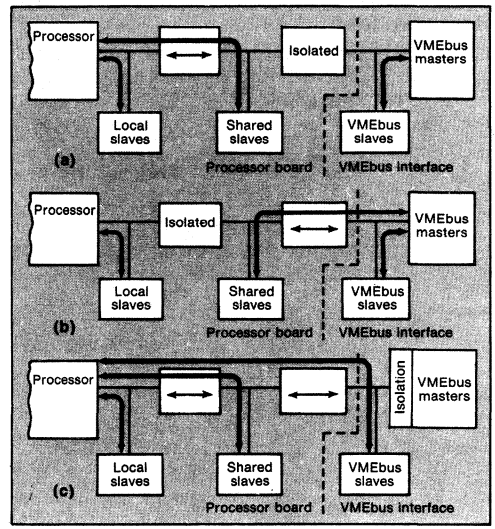
In all configurations, the bus controller is clocked at up to 25 MHz by an external signal derived from TTL chips or an on-chip crystal oscillator. Typical propagation time through the controller is 75 ns. However, not all of that time is factored into determining system speed, because up to 35 ns of the delay can be overlapped with address decoding.

Coordinating data transfers

The control circuit proper—including local and VMEbus address-decoding logic, bus request lines, and master and slave control blocks—is basically driven by inputs from four lines: the VMEbus Address Strobe (AS), and the on-board master's Address Strobe (MAS). Address-decoding control lines SLV, ONBD and VME indicate whether each cycle begun by these strobes accesses a shared on-board slave or a slave on another VMEbus board (Fig. 3). (The table defines all the bus controller's signals.)

To ensure glitch-free switching of its asynchronous inputs, the controller must give the sampled strobes enough time to settle (normally required in state machines of its type) before acting on it. During that settling time, addresses may be decoded. To be more specific, the address-decoding inputs need not be valid until shortly before the clock edge that follows the edge on which the address strobe first goes low. Thus, for a 25-MHz clock rate, the address-decoding lines need not be valid until 30 ns after the leading edge of the strobe. On the VMEbus, the minimum address time at a slave is 10 ns; thus there is a maximum delay of 40 ns through the address decoder—sufficient time for virtually all decoding schemes. If, for some reason, that decoding time is insufficient, slowing the clock affords even more leeway.

The interbus switching logic remains in the same state until a request cycle causes it to switch control of the shared bus. This scheme



2. In the most typical state of non-VMEbus operation, the controller lets the processor access its on-board resources on either a local or shared bus, while other masters and slaves communicate on the VMEbus (a). In a slave state, other bus masters may use the shared bus, but the processor can use only its local bus (b). The master state allows the processor to use any bus and thus communicate with local, shared, or VMEbus slaves; other VMEbus masters can access only their on-board slaves (c).

provides minimum timing overhead for master-only and slave-only applications, as well as for consecutive master or slave cycles in a master-slave design. If AS and MAS both go low within the same clock period and both addresses point to the shared bus, the bus controller will grant access to the one that used the bus most recently. The other bus will assume control after the opposing address strobe has been released. Thus neither side can hog the shared bus. Similarly, if a request cycle arrives from one side during execution of a cycle on the other side, the first side will capture the next access to the shared bus.

The primary difference between the processor and DMA versions of the controller lies in the allocation of the decoding inputs associated with MAS. In the processor version, the two associated decoding inputs, ONBD and VME, enable the controller to differentiate among processor cycles on the local, shared, or VMEbus structures. The chip will ignore cycles on the local bus; in fact, those cycles can proceed simultaneously with cycles initiated by another VMEbus master on the shared bus (see Fig. 2b again).

In the DMA version, one of the two address-decoding inputs, LBR, serves as a local bus

The SBC68172 bus controller's signals

Pinout	Input or output	Description
AS	Both	VMEbus address strobe
BBSY	Output	Indicates that controller has been granted a VMEbus request
BGIN	Input	Bus grant input
BGOUT	Output	Bus grant output
BR	Output	Bus request sent to the central system's arbiter
DDIR, DEN	Output	Direction and enable controls, respectively, for data transceivers linking the shared bus to the VMEbus
DSEN	Output	Enables VMEbus data strobes during a VMEbus master cycle
DSI	Input	Logical OR of shared (on-board) data strobes
DTACK, BERR	Both	VMEbus slave response signals
LDTACK, LBERR	Both	On-board slave responses, these lines drive or are driven by DTACK or BERR.
LBG	Output	Grants local bus to DMA controller (DMA version only)
LBR	Input	Local bus request from DMA controller (DMA version only)
MAS	Input	Onboard master's address strobe
MASTEN	Output	Connects processor's local bus to the shared bus; enables drivers for the Address, Read/Write, and Data Strobe signals; controls direction for address transceivers
ONBD	Input	Indicates whether the current processor or DMA controller is accessing a slave on the shared bus
RELSE	Input	Indicates that on-board logic wants to release control of the VMEbus
RESET	Input	Clears bus controller's logic
R/W	Input	Shared (on-board) Read/Write signal; low for write operations
SLV	Input	Address-decoding line from the VMEbus
SLVSEL	Output	Indicates that shared slaves have been selected for the current cycle
V_{CC}, V_{SS}	—	Power and ground leads, respectively
VME	Input	Indicates that the processor is accessing a slave on the VMEbus board (processor version only)
VMEEN	Output	Connects the shared bus to the VMEbus
$XTAL_1$, $XTAL_2$	Input	Accepts crystal or TTL-level clock

request signal that causes the bus controller to call for the VMEbus, assuming the chip is not controlling it already. That is, the DMA controller must gain control of the VMEbus before performing any operation. When the request for a system bus is granted, the DMA controller can proceed, assuming AS and MAS are high. In many applications, the DMA controller's Bus Grant Acknowledge output (BGACK) can be connected to the bus controller's Release input (RELSE), enabling the DMA controller to throttle and control its use of the VMEbus.

Request logic

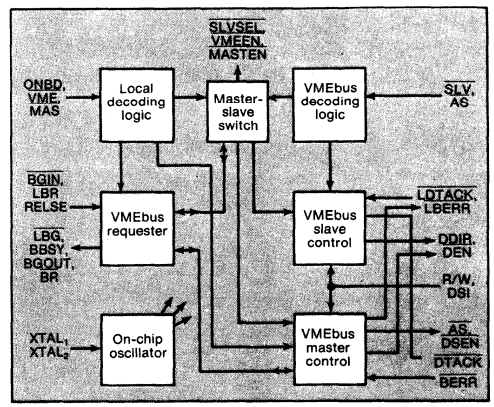
Though related, the VMEbus request logic differs from the master-slave switching logic that controls access to the shared bus. To gain access to the VMEbus, the controller must deal with the central system arbiter. With MAS low, the controller waits for the arbiter to grant it the bus, in the meantime remaining open so that masters and slaves can communicate. If the bus grant signal arrives while a slave cycle is in progress, the controller awaits the cycle's completion; otherwise, it activates the drivers that link the local to the shared bus. As soon as any current VMEbus cycle is done, the bus controller activates the drivers that link the shared bus to the VMEbus, and data transfer begins.

Once the bus controller has won control of the VMEbus, it relinquishes it only when a RELSE signal arrives. That signal increases the bus controller's flexibility, so that it can release a bus to another master when required, or after a given number of cycles. The controller can also release control during a VMEbus cycle or between cycles.

Since the controller retains the present switch direction until forced to change, SLV, ONBD and VME must be decoded from the local bus and VMEbus rather than from the shared bus. Other control signals such as Read/Write and the data strobes are taken from the shared bus, thereby minimizing the bus controller's pin count. The bus controller does not interface directly with the data strobes and is processor-independent in the sense that it can handle one, two or several data strobe signals.

The bus controller goes into action when an address strobe (AS or MAS) arrives. One clock period after AS or MAS goes low, the controller's associated address-decoding inputs (SLV for AS, ONBD and VME for MAS) are sampled. The ONBD and VME signals must be held valid until shortly after MAS goes high.

The SLV signal must be held valid until shortly after the next low-going edge of the Data Acknowledge (DTACK) and Bus Error (BERR) signals. With that timing scheme, the bus controller can take advantage of the fact that a new address and address strobe can be presented on the VMEbus while a preceding cycle is being completed (address look-ahead



3. Inside the bus controller chip are local bus and VMEbus address decoding logic, bus request lines, and master-slave control blocks. The on-chip oscillator drives the clock input for the other blocks.

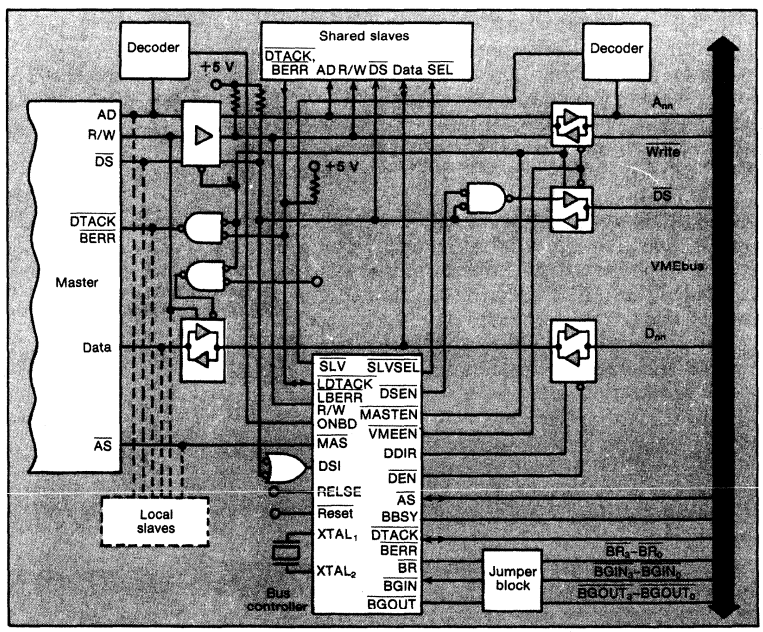
or pipelined operation). It also avoids problems in coping with the fact that a bus master can withdraw the address, address strobe, and data strobes in any order at the end of a given cycle.

What action the controller takes depends on how competing cycles are decoded. For instance, if SLV is sampled high after AS drops low, or if ONBD is sampled low and VME high after MAS goes low, the controller will ignore the cycle. If ONBD is sampled high after MAS goes low and the master-slave switch is in the slave position, the chip waits for the current VMEbus cycle to finish. Thereafter, it disables the transceivers for the VMEbus's address lines, read/write lines, and data strobes by driving the VMEbus Enable output (VMEEN) high. Then the Master Enable line (MASTEN) is moved low, transferring the local bus address to the shared bus.

If, on the other hand, SLV is brought low after AS and the master-slave switch is in the master position, the chip will wait for any communication between the on-board master and shared slaves to end before bringing MASTEN high and VMEEN low, thereby placing the VMEbus address on the shared bus.

One clock period after sending the proper address to the shared bus, the controller indicates the selection of a shared slave by driving the Slave Select output (SLVSEL) low. SLVSEL is released shortly after AS or MAS goes high, but MASTEN and VMEEN are held in their present logic states until forced to change.

The controller's arbiter comes into play when the on-board master initiates a cycle with a VMEbus slave, i.e. such as when VME is sampled low after MAS goes low. If the controller chip does not have control of the VMEbus, it drives the Bus Request line (BR) low to request the bus. Once the appropriate Bus Grant In (BGIN) goes low, the bus controller assumes control of the VMEbus and an-



4. Two versions of the bus controller can work with either a master processor or a DMA controller. In the former case, shown here, additional parts are needed to isolate the processor from its shared slaves.

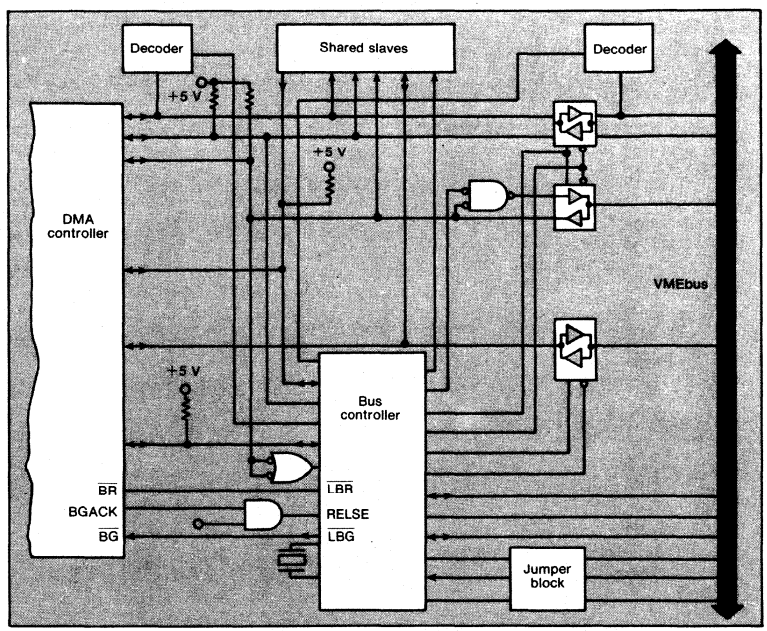
nounces its takeover by bringing the Bus Busy line ($\overline{BBSY}$) low.

If the master-slave switch is in the slave position, the controller waits for any current VMEbus slave operation to end, represented by $\overline{AS}$ going high and the Data Strobe Input ($\overline{DSI}$) going low. $\overline{MASTEN}$ and $\overline{VMEEN}$ are then brought low and high, respectively. Once the current VMEbus cycle is complete, the controller drives $\overline{VMEEN}$ low, and Data Enable ($\overline{DEN}$) low when appropriate, permitting shared signals out on the VMEbus. After two clock periods, the controller is ready to bring $\overline{AS}$ and the Data Strobe Enable signal ($\overline{DSEN}$) low to begin a new VMEbus cycle.

If the bus controller retains the VMEbus until the onboard master initiates another cycle, it assumes that the address setup time has been met. It then drives $\overline{AS}$ and $\overline{DSEN}$ low immediately after sampling ONBD and $\overline{VME}$.

Some practical designs

The bus controller can be used in four distinct systems: in processor-based master-slave designs, for master-slave operation with a DMA controller, and in master-only or slave-



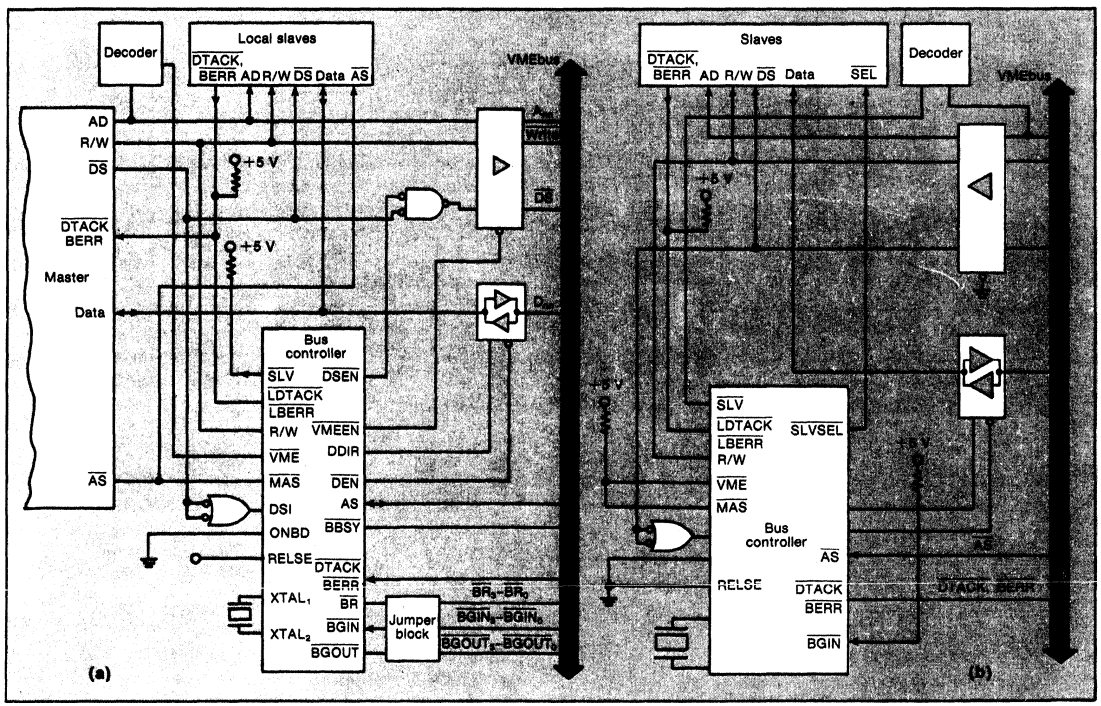
5. When used with the bus controller in a master-slave configuration, the DMA controller is connected directly to the shared bus. Most of the other connections linking the bus controller and shared slaves to the VMEbus are identical to those of the general processor-based system shown in Fig. 4.

only arrangements. The first setup is the most complex (Fig. 4), but also represents the most general case. In that design, MAS is taken from the local bus, and the ONBD and VME decoding lines is decoded from the local bus address. Similarly, AS is directly connected to the bus controller, and SLV works on the VMEbus address.

The local bus Address (AD), Read/Write (R/W), and Data Strobe (DS) signals are presented to the shared bus when MASTEN is low. For 24-bit processors, four 74F244s can serve as the signal drivers, and 74F545s can be used as data transceivers between the local and shared buses. The local bus's R/W line can act as the bus direction control. MASTEN must be included in the enabling inputs of these transceivers through 74F32 AND gates.

The VMEbus address and Write signals, also are connected to the shared bus through transceivers, with MASTEN determining the direction of the data flow and VMEEN supplying the enable signal. Here again, 74F545 transceivers can be used, but MASTEN must be inverted.

The bus controller works in a similar manner with the VMEbus and a DMA controller for master-slave applications (Fig. 5). In operation, the DMA controller first requests control of the bus, waits for control to be granted, and finally enables itself onto the shared bus. Since the DMA controller must be programmed, it is connected directly to the shared bus, rather than being isolated from the shared slaves as in the processor-based master-slave application. Virtually all other



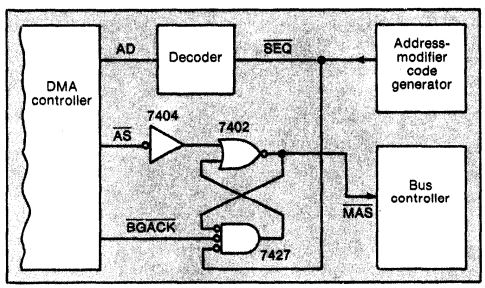
6. The bus controller can easily handle either master-only (a) or slave-only (b) applications. Most of the circuit connections remain the same for both configurations. Note that no address-decoding is required for master-only purposes.

pin connections are the same.

Finally, the bus controller can work in either master-only or slave-only systems. When it is interfacing a master processor to the VMEbus (Fig. 6a), no VMEbus address decoding is required. Instead, the chip's SLV input is tied high, and the master-slave switch remains in the master position. Because MASTEN always remains low, it plays no part in the controller's interface logic. Also, AD, Write, and the data strobes can use simple drivers like 74F244 buffers instead of transceivers.

Handling simpler tasks

For slave-only applications, the VMEbus address-decoding is connected to SLV (Fig. 6b). But since no on-board decoding is required, VME can be tied high and ONBD grounded. After a system reset, following the first time the slaves have been accessed, MASTEN stays high and VMEEN remains low (neither signal is useful in the interface logic). The AD, Write, and DS lines can be used to drive simple receivers, such as an 74S244. Unless there is a special design consideration, the receivers can be enabled all the time, so that the VMEbus address is always brought on board.



7. A simplified circuit allows VMEbus transfers to be processed sequentially. The operation occurs whenever the DMA controller accesses a user-defined range of addresses over the VMEbus.

In operation, the bus controller provides reliable address decoding, controls the VMEbus data transceivers (74F545s) via the DDIR and DEN lines, and drives DTACK and BERR in accordance with the VMEbus protocol. Again, many of the connections found in the master-slave circuit are common to the slave-only circuit; runs from unlabeled pins on the chip and other IC blocks indicate an identical connection in the master-slave circuit.

Stepping through data

One of the most powerful capabilities of the controller is its ability to transfer a block of data along the VMEbus by means of the bus's sequential, or ascending-access, mode. In operation, a VMEbus master places a starting address on the bus with an address-modifier code that specifies sequential data transfer. All slaves capable of sequential operation latch that address into on-board counters.

Thereafter, the master holds its AS line low during multiple data transfers, each of which is signaled by data strobes. The slaves increment the address counters on each transfer and can pipeline transfers to save still more time.

A simple circuit illustrates how the bus controller can work with a DMA controller in transferring blocks of data (Fig. 7). If the DMA controller generates a set of addresses in a predefined range of VMEbus addresses, the decoder will bring its sequence line (SEQ) low, causing a sequential address-modifier code to be placed on the bus.

The states of SEQ and BGACK keep MAS low between address and data strobes from the DMA controller. Thus, the bus controller holds AS low on the VMEbus as long as sequential operation is desired. Speed rises because the VMEbus slave can run ahead of the DMA controller and prefetch data for subsequent read operations.□

BIPOLAR 8-BIT MICROPROCESSOR FAMILY

Standard products

BIPOLAR MICROPROCESSORS

Originally published by Signetics January 1984

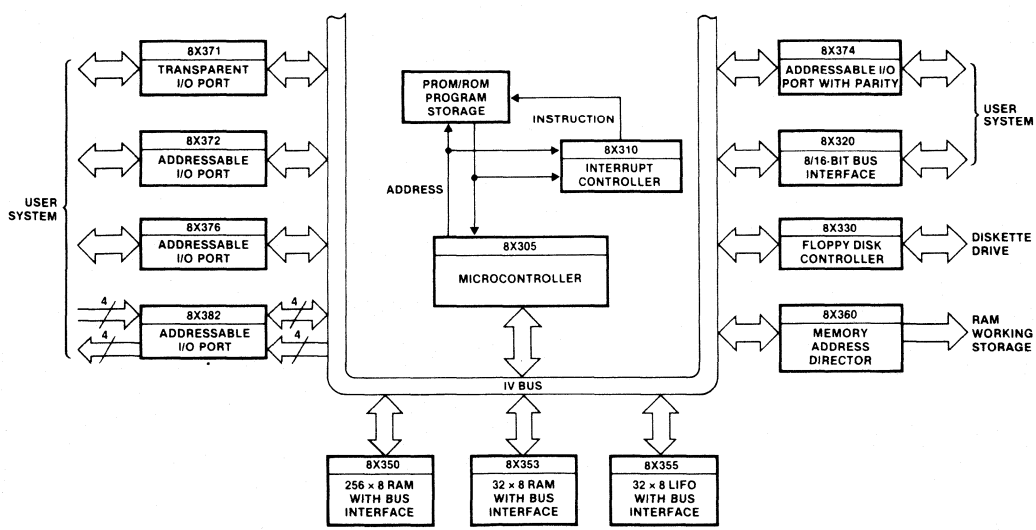
FEATURES

- Minimum Parts Count
- Bipolar Device Using Low Power Schottky Technology
- High Performance
- Source/Destination Architecture
- Design Flexibility
- After-Sales Support

USER BENEFITS

- Reduced Cost
- Extended Life of Product
- Full System Drive Without Buffering
- TTL Compatibility
- Proven Reliability
- Instruction Cycle Time of 200ns (8X305) or 250ns (8X300)
- Bit Addressability — Instruction Data Formats of 1-to-8 Bits Without Added Delay
- Efficient Use of Fixed Instruction Set
- Easy to Program
- Well-Suited for Control Applications
- Single-Chip Processor
- Full Complement of Support Devices
- Development System
- Training
- Complete Documentation
- Applications Support from Qualified Field Engineers

8X300 FAMILY



NOTES
1. The shaded devices are designed to take advantage of the improved performance of the 8X305.
2. Refer to "Reference Table" on next page for a descriptive summary of each part in the 8X300 Family.
3. The actual parts used in a specific system are selected based on the application requirements.

PRODUCT LINE OVERVIEW

The Bipolar LSI Microprocessor product line is centered around two high-speed, Schottky-fabricated, MicroControllers and a large family of I/O devices, support ICs, and development support tools to expedite and simplify system design. No other microprocessor product line offers the advantages of the 8X300 Family in systems requiring intelligent control of high-speed data streams.

The MicroControllers themselves are capable of fetching, decoding, and executing each instruction in one machine cycle (250ns for the 8X300 and 200ns for the 8X305). A single instruction can read data from the bus, mask it, shift it, perform an ALU operation on it, rotate it, merge it, and return it to the bus. The architecture and instruction set of each processor are designed to provide high data throughput with both bit and byte oriented operations. The 8X305 offers more on-chip registers and significant data handling capability improvements over the original 8X300.

Because of its extremely high speed, the 8X300 family is able to perform through software many operations that would otherwise require additional hardware in the system. For example, using either MicroController, the 8X330 Floppy Disc Controller, and other support devices, a complete diskette drive controller can be built using only 10 Integrated Circuits. The resulting controller is programmable and capable of supporting multiple drive types and formats. Low parts counts typical of 8X300 Family designs result in reduced assembly and testing time, lower power consumption, and improved reliability.

The 8X300 Family is implemented using bipolar, Low-power Schottky technology to provide TTL speeds and drive capability without additional buffering. In addition, the family is compatible with most special-purpose devices often required in control applications. The excellent environmental characteristics of the technology make the 8X300 family ideal for military applications as well.

A complete complement of development support tools are offered to simplify design using the 8X300. A self-contained universal development system is available that supports full speed in-circuit emulation. Software tools are provided to enable use of mainframe or minicomputers to generate 8X300 software. Complete documentation is in place, including both data sheets and comprehensive reference manuals. In addition, evaluation and demonstration systems are available. To supplement these tools, Signetics employs a large, professional staff of applications engineers both in the field and at the factory to support your 8X300 design efforts.

8X300 FAMILY REFERENCE TABLE

PART NO.	PRODUCT DESCRIPTION	FUNCTION
8X300	MicroController	250ns processor for I/O and data control
8X305	MicroController	200ns processor for I/O and data control
8X310	Interrupt Controller	3 prioritzed interrupts with 4-level stack
8X320	Bus Interface Array	2-port RAM for 8/16-bit mailbox interfacing
8X330	Floppy Disk Controller	1Mb/sec data rate, programmable, supports ECC
8X350	Bipolar RAM	256 x 8 high-speed memory with bus interface
8X360	Memory Address Director	16-bit address controller for working storage
8X337	Transparent I/O Port	High-speed, 8-bit bidirectional, 3-state
8X372	Addressable I/O Port	High-speed, 8-bit bidirectional, synchronous, 3-state
8X376	Addressable I/O Port	High-speed, 8-bit bidirectional, asynchronous, 3-state
8X382	Addressable I/O Port	High-speed, 4-in/4-out, 3-state
8T31	Transparent I/O Port	8-bit bidirectional, 3-state
8T32	Addressable I/O Port	8-bit bidirectional, synchronous, 3-state
8T36	Addressable I/O Port	8-bit bidirectional, asynchronous, 3-state

MICROCONTROLLER

Originally published by Signetics January 1984

FEATURES

- **Fetch, Decode, and Execute a 16-bit instruction in a minimum of 250-nanoseconds (one machine cycle)**
- **Bit-oriented instruction set (addressable single-or-multiple bit subfields)**
- **Separate address, instruction, and I/O buses**
- **Source/destination architecture**
- **On-Chip oscillator and timing generation**
- **Eight 8-bit working registers**
- **TTL inputs and outputs**
- **BiPolar Low-Power Schottky technology**
- **3-State I/O bus**
- **Single +5V supply**

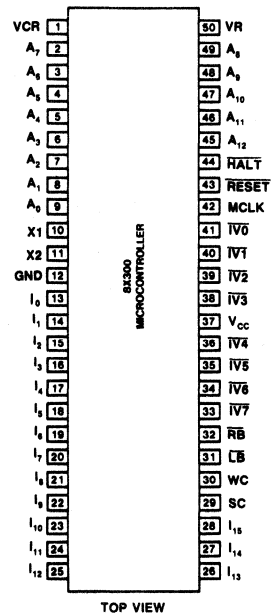
ARCHITECTURAL OVERVIEW

The Signetics 8X300 Microcontroller (Figure 1) is a high-speed bipolar microprocessor implemented with low-power Schottky technology. The 8X300 brings together all the qualities needed—SPEED, FLEXIBILITY, and ECONOMY—for systems design in the many areas that require reliable bit stream management. Consider!—5V operation, TTL bus compatibility, and an on-chip clock—the result, a system with fewer parts. Consider!—the inherent power of LSI logic (programmable Rotate, Mask, Shift, and Merge functions in the data-processing path) and the ability to Fetch, Decode, and Execute a 16-bit instruction in a minimum of 250-nanoseconds—the result, a system with superior bit handling capabilities. Consider!—the 250ns cycle time in conjunction with extended microcode—the result, the flexibility of bit-slice devices with the programming ease of MOS microprocessors. Now, consider the results!—a device tailored to bit-stream management in the areas of Industrial Control, Input/Output Control, and Data Communications.

The 8X300 uses three separate buses—one for 13-bit instruction addresses, one for 16-bit instructions, and a bidirectional 8-bit input/output data bus; except for the I/O bus, there are no time multiplexing of functions.

PIN CONFIGURATION

I PACKAGE



TOP VIEW

ORDER NUMBERS

N8X300I, S8X300-1, S8X300-2

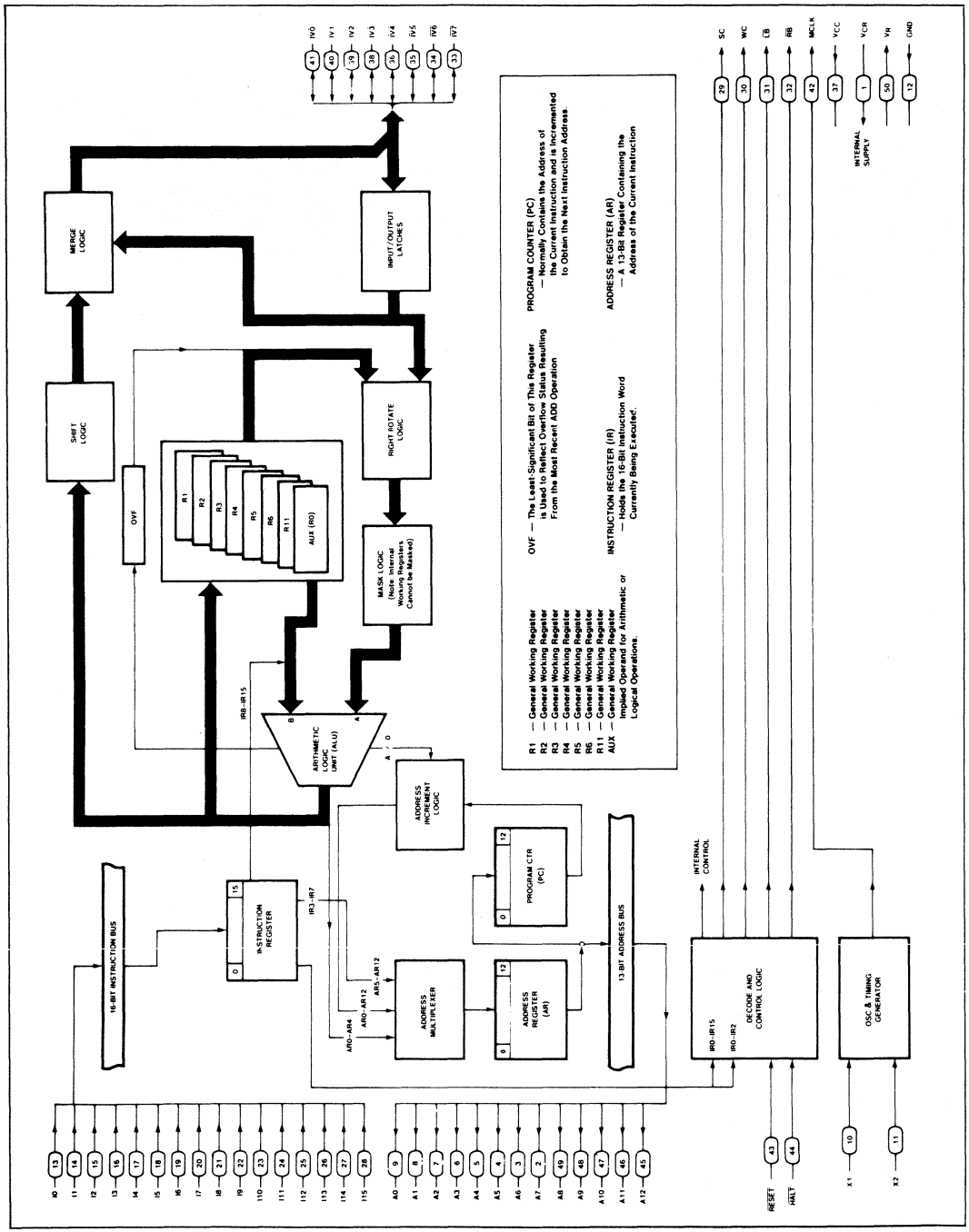
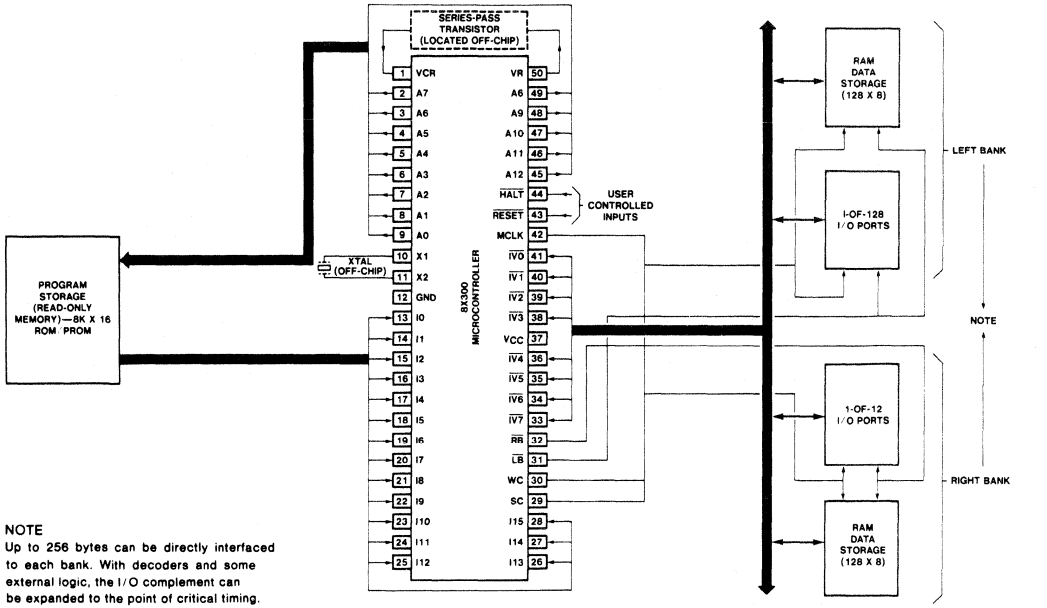


Figure 1. CPU Architecture and PIN Designations For 8X300 Microcontroller



PIN NO.	IDENTIFIER	NAME AND FUNCTION	ACTIVE STATE
2-9/45-49	A0-A12	**Program Address Lines:** These outputs permit direct addressing of up to 8192 words of program storage. A high voltage level equals a binary "1"; A12 is Least Significant Bit.	High
13-28	I0-I15	**Instruction Lines:** These input lines receive 16-bit instructions from program storage. A high voltage level equals a binary "1"; I15 is Least Significant Bit.	High
33-36 38-41	$\overline{IV0}$-$\overline{IV7}$	**Input/Output Bus:** These bidirectional three-state lines communicate with up to 512 I/O devices (256 per bank). A low voltage level equals a binary "1"; $\overline{IV7}$ is Least Significant Bit.	Low
10 & 11	X1 & X2	**Connections** for a capacitor, a series-resonant crystal, or an external clock source with complementary outputs. For precise frequency control, a crystal or external source is required.	—
42	MCLK	**Master Clock:** This output is used for clocking I/O devices and/or synchronization of external logic.	High
30	WC	**Write Command:** When signal is high (binary 1), data is being output on pins $\overline{IV0}$-$\overline{IV7}$ of I/O bus.	High
29	SC	**Select Command:** When signal is high (binary 1), an address is being output on pins $\overline{IV0}$-$\overline{IV7}$ of I/O bus.	High
31	$\overline{LB}$	When the $\overline{LB}$ signal is low (binary 0), any one of up-to-256 I/O devices (or memory locations) in the left bank can be accessed. When the address of a particular device (or memory location) matches the address on the IV bus, that particular device (or memory location) is enabled and selected for input/output operations. All addresses on the left bank that do not match are deselected.	Low
32	$\overline{RB}$	When the $\overline{RB}$ signal is low (binary 0), any one of up-to-256 I/O devices (or memory locations) in the right bank can be accessed. When the address of a particular device (or memory location) matches the address on the I/O bus, that particular device (or memory location) is enabled and selected for input/output operations. All addresses on the right bank that do not match are deselected.	Low
43	$\overline{RESET}$	When reset input is low (binary 0), the microcontroller is initialized—sets Program Counter/Address to zero and inhibits MCLK output.	Low
44	$\overline{HALT}$	When halt input is low (binary 0), internal operation of microcontroller stops at the start of next instruction. The stop function does not inhibit MCLK or affect any internal registers.	Low
50	VR	Internally-generated reference output voltage for external series-pass transistor.	—
1	VCR	Regulated voltage input from series-pass transistor (2N5320 or equivalent).	—
12	GND	Circuit ground.	—
37	V_{CC}	Input connection for +5V power.	—

Figure 2. Typical 8X300 System with Pin Definitions

TYPICAL 8X300 SYSTEM HOOKUP

Although the system hookup shown in Figure 2 is of the simplest form, it provides a fundamental look at the 8X300 microcontroller and peripheral relationships. As indicated, program storage can be either ROM or PROM and, by using various addressing-methods/decoding-schemes, memory paging techniques can be easily implemented. Also, by proper bit assignment, some external interface logic and, under software control, the program memory can be used as a storage device for interrupt-service subroutines. The user interface ($\overline{IV0}$ through $\overline{IV7}$) is capable of addressing 256 Input/Output ports and, with the additional bank-select bit ($\overline{LB}$ and $\overline{RB}$), the number of addressable I/O ports is 512—the left bank and right bank each consisting of 256 ports. The I/O ports of each bank can be used in a variety of ways; one of these ways is shown in Figure 2. When $\overline{LB}$ is active low, the left bank can be enabled and, providing there is an address match, anyone of 128 I/O ports or anyone of 128 locations within the RAM memory can be accessed for input/output operations. When $\overline{RB}$ is active low, the same set of conditions are applicable to the right bank. With some sacrifice in speed, any given I/O port can be interfaced to a memory peripheral or other I/O device of the user.

PROGRAM STORAGE INTERFACE

As shown in Figure 2, program storage is connected to output address lines A0 through A12 (A12 = LSB) and input instruction lines I0 through I15. An address output on A0/A12 identifies one 16-bit instruction word in program storage. The instruction word is subsequently input on I0/I15 and defines the microcontroller operations which are to follow.

The Signetics 82S115 PROM or any TTL-compatible memory can be used for program storage. (Note. The worst-case access time depends upon the instruction cycle time, and also, the overall system configuration.)

I/O INTERFACE AND CONTROL

An 8-bit I/O data bus is used by the microcontroller to communicate with two fields of I/O devices. The complementary $\overline{LB}$ and $\overline{RB}$ signals identify which field of the I/O devices is enabled.

Both data and address information are output on the I/O bus. The SC (Select Command) and WC (Write Command) signals distinguish between data and address information as follows:

SC	WC	FUNCTION
High	Low	I/O address is being output on the I/O (IV) bus
Low	High	I/O data is being output on the I/O (IV) bus
Low	Low	Input data expected from selected I/O device
High	High	Invalid (not generated by 8X300)

DATA PROCESSING

From a data processing point of view, the 8X300 microcontroller chip (Figure 1) contains eight 8-bit working registers (R1 through R6, R11, AUXiliary), an arithmetic logic unit (ALU), an overflow register (OVF), rotate/shift/mask/merge logic, and a bidirectional 8-bit I/O bus. Internal 8-bit data paths connect the registers and I/O bus to the ALU inputs, and the ALU output to the registers and I/O bus. Inputs to the ALU are preceded by the data-rotate and data-mask-logic and the ALU output is followed by the shift and merge logic. Any one or all of the logic functions can operate on 8-bits of data in a single instruction cycle. Data from the source register can be right-rotated (end around) before processing by the ALU; external data (I/O bus) can also be masked to isolate a portion of the 8-bit field. Since the ALU always processes 8-bits of data, bit positions not specified by the mask operation are filled with zeroes.

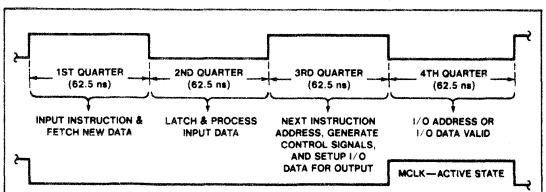
When less than 8-bits of data are specified as output to the I/O bus from the ALU, the data field (shifted and masked, as required) is merged with prior contents of the I/O latches to form the output data. Bit positions of the I/O data not affected by the logic operations are not modified. Depending upon whether an I/O peripheral or an internal register is specified in the instruction as the source of data, the I/O latches contain, respectively, I/O-bus source data or destination data. For instance, when an internal register is specified as a source of data and an I/O peripheral as the destination, data from the peripheral is read into the I/O latches at the start of the instruction cycle; processed data is then merged with contents of the I/O latches to form the I/O output data at the end of the instruction cycle. When an I/O peripheral is specified as both data source and destination, data from the source is used both as the input to the I/O latches and as data to be processed; the processed data is then merged with data from the I/O latches to form the previously-described I/O bus output. If the data source and destination are on opposite banks of the 8X300 bus, the destination data is written with a full 8-bits, since the prior contents were not stored in the I/O latches.

INSTRUCTION CYCLE

Each microcontroller operation is executed in a single instruction cycle. The instruction cycle is divided into quarters with each quarter cycle being as short as 62.5-nanoseconds. Figure 3 shows the general functions that occur during each quarter cycle; specifics regarding minimum/maximum timing and other critical values are described under "Design Parameters" in this data sheet. During the first quarter cycle, a new instruction from program storage is input on signal lines I0 through I15; simultaneously, new data is fetched via the input/output bus ($\overline{IV0}$ through $\overline{IV7}$). At the end of the first quarter cycle, the new instruction is latched in the instruction register and the new I/O data is present at the input of the chip but is not, as yet, latched by the IV latches.

In the second quarter cycle, the I/O data stabilizes and preliminary processing is completed; at the end of this quarter, the IV latches are closed and final processing can be accomplished. During the third quarter cycle, the address for the next instruction is output to the I/O (IV) bus, control signals are generated, and I/O data is setup for the output

phase. During the fourth quarter cycle, a master clock signal (MCLK) generated by the 8X300 is used to latch valid address or data into peripheral devices connected to the IV bus; MCLK is also used to synchronize any external logic with timing circuits of the 8X300. To summarize the action, the first half of the instruction cycle deals primarily with input functions and the second half is mostly concerned with output functions.



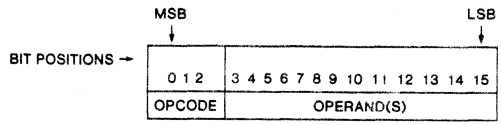
NOTES
1. New instruction must be accepted and latched at end of first quarter cycle.
2. The I/O data latches are open for the first two quarter cycles, that is, for 125-nanoseconds.
3. The address is output during third quarter cycle.
4. I/O bus drivers are active (turned on) during third and fourth quarter cycles.

Figure 3. Instruction Cycle and MCLK with: Crystal = 8MHz and Cycle Time = 250 ns

INSTRUCTION SET

General Format and Basic Operations

The 16-bit instruction word (I0 through 15) from program storage is input to the instruction register (Figure 1) and is subsequently decoded to implement the events to occur during the current instruction cycle. The instruction word is formatted as follows:



Rather than discrete instructions, the three operation code (OP CODE) bits specify eight instruction classes. Each instruction class is subject to a number of powerful variations; these variations are specified by the thirteen operand bits. General areas of control for the eight instruction classes are:

- Arithmetic and Logic Operations (ADD, AND, AND XOR)
- Movement of Data and Constants (MOVE and XMIT)
- Branch or Test (JMP, NZT, and XEC)

Basic operations for each of the eight instruction classes are as follows; a summary of the instruction set is provided in Table 1.

MOVE—data in source register or I/O-bus input is moved to destination register or I/O-bus output. Data can be shifted any number of places and/or masked to any length.

ADD—data in source register or I/O-but input is added to content of AUX (R0) register and the result is placed in the destination register or I/O-bus output. Data can be shifted and/or masked, as required.

Table 1. SUMMARY OF 8X300 INSTRUCTION SET

INSTRUC CLASS	OPCODE	FORMATS	DESCRIPTION	I/O CONT SIG	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE: INPUT PHASE (INSTRUCTION INPUT & DATA PROCESSING)	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE: OUTPUT PHASE (ADDRESS & I/O BUS)
MOVE	0	**F1: Register to Register** 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | R | D Invalid values of "S": 07_8, 17_8, 20_8-37_8 Invalid values of "D": 10_8, 20_8-37_8	(S) → D Move content of internal register specified by S-field to internal register specified by D-field. Prior to the "MOVE" operation, right-rotate contents of internal source register by octal value (0 through 7) defined by the R-field.	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L X X	H if "D" = 07_8, 17_8 L H if "D" = 17_8 L if "D" = 07_8
		F2: I/O Bus to Register 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | L | D Valid values of "S": 20_8-37_8 Invalid values of "D": 10_8, 20_8-37_8	Move right-rotated I/O bus (source) data specified by the S-field to internal register specified by the D-field. The L-field specifies the length of source data starting from the LSB-position and, if less than 8-bits, the remaining bits are filled with zeroes.	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L L if "S" = 20_8-27_8 H if "S" = 30_8-37_8	H if "D" = 07_8, 17_8 L H if "D" = 17_8 L if "D" = 07_8
		F2: Register to I/O Bus 0 1 2 3 | 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | L | D Invalid values of "S": 07_8, 17_8, 20_8, 37_8 Valid values of "D": 20_8-37_8	Move contents of internal register specified by the S-field to the I/O latches. Before outputting on I/O bus, data is shifted as specified by the least significant octal digit of the D-field and the bits specified by the L-field are merged with the latched I/O data.	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L L if "D" = 20_8-27_8 H if "D" = 30_8-37_8	L H L if "D" = 20_8-27_8 H if "D" = 30_8-37_8
		F2: I/O Bus to I/O Bus 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | L | D Valid values of "S": 20_8-37_8 Valid values of "D": 20_8-37_8	Move right rotated I/O-bus (source) data specified by the S-field to the I/O latches. Before outputting on I/O bus, shift data as specified by the D-field; then merge source and latched I/O data as specified by the L (length) field.	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L L if "D" = 20_8-27_8 H if "D" = 30_8-37_8	L H L if "D" = 20_8-27_8 H if "D" = 30_8-37_8

Table 1. SUMMARY OF 8X300 INSTRUCTION SET (Continued)

INSTRUC CLASS	OPCODE	FORMATS	DESCRIPTION	I/O CONT SIG	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE	
					INPUT PHASE (INSTRUCTION INPUT & DATA PROCESSING)	OUTPUT PHASE (ADDRESS & I/O BUS)
ADD	1	Same as MOVE instruction class	(S) plus (AUX) → D Same as MOVE instruction class except that contents of AUX (R0) register are ADDed to the source data. If there is a "carry" from MSB, then OVF (overflow) = 1, otherwise OVF = 0.	Same as MOVE instruction class		
AND	2	Same as MOVE instruction class	(S) ∧ (AUX) → D Same as MOVE instruction class except that contents of AUX (R0) register are ANDed with source data.	Same as MOVE instruction class.		
XOR	3	Same as MOVE instruction class	(S) ⊕ (AUX) → D Same as MOVE instruction class except that contents of AUX (R0) register are exclusively ORed with source data.	Same as MOVE instruction class.		
XEC	4	**F3: Register Immediate** 0 1 2 | 3 4 5 6 7 | 8 9 10 11 12 13 14 15 OPCODE | S | J Invalid values of "S": 07_8, 17_8, 20_8-37_8 Valid values of "J": 000_8-377_8	**Execute instruction at current page address offset by J (literal) + (S). Return to normal instruction flow unless a branch is encountered.** Execute instruction at an address determined by replacing the low-order 8-bits of the Program Counter with the following derived sum: • Value of literal (J-field) plus • Contents of internal register specified by S-field The PC is not incremented and the overflow status (OVF) is not changed.	SC = WC = $\overline{LB}$ =	L L X	L L X
		F4: I/O Bus Immediate 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | L | J Valid values of "S": 20_8-37_8 Valid values of "J": 00_8-37_8	Execute instruction at an address determined by replacing the low-order 5-bits of Program Counter with the following derived sum: • 5-bit value of literal (J-field) plus • Value of rotated source data specified by S-field (The L-field specifies the length of source data starting from the LSB-position and, if less than 8-bits, the remaining bits are filled with zeros; the Program Counter is not incremented and the overflow status (OVF) is not changed.)	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L L if "S" = 20_8-27_8 H if "S" = 30_8-37_8	L L X X
NZT	5	**F3: Register Immediate** 0 1 2 | 3 4 5 6 7 | 8 9 10 11 12 13 14 15 OPCODE | S | J Invalid values of "S": 07_8, 17_8, 20_8-37_8 Valid values of "J": 000_8-377_8	**If data specified by the S-field is not equal to zero, jump to current page address offset by value of J-field; otherwise, increment the Program Counter.** If contents of internal register specified by S-field is non-zero, transfer to address determined by replacing the low-order 8-bits of Program Counter with "J", otherwise, increment PC.	SC = WC = $\overline{LB}$ =	L L X	L L X
		F4: I/O Bus Immediate 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | L | J Valid values of "S": 20_8-37_8 Valid values of "J": 00_8-37_8	If right-rotated I/O bus data is non-zero, transfer to address determined by replacing low-order 5-bits of Program Counter with "J", otherwise, increment PC. (The L-field specifies the length of source I/O data starting from the LSB-position and, if less than 8-bits, the remaining bits are filled with zeroes.)	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L L if "S" = 20_8-27_8 H if "S" = 30_8-37_8	L L X X
XMIT	6	**F3: Register Immediate** 0 1 2 | 3 4 5 6 7 | 8 9 10 11 12 13 14 15 OPCODE | D | J Invalid values of "D": 10_8, 20_8-37_8 Valid values of "J": 000_8-377_8	Transmit J → D Transmit and store 8-bit binary pattern in J-field to internal register specified by D-field.	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L X X	H if D = 07_8 or 17_8 L H if D = 17_8 L if D = 07_8
		F4: I/O Bus Immediate 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | D | L | J Valid values of "D": 20_8-37_8 Valid values of "J": 00_8-37_8	Transmit binary pattern in J-field to I/O bus. Before putting data on I/O bus, shift literal value "J" as specified by the D-field and merge bits specified by the L-field with existing I/O bus data. If the L-field specifies more than 5-bits starting from the LSB-position, all remaining bits are set to zero.	SC = WC = $\overline{LB}$ = $\overline{LB}$ =	L L L if D = 20_8-27_8 H if D = 30_8-37_8	L H L if D = 20_8-27_8 H is D = 30_8-37_8
JMP	7	**F5: Address Immediate** 0 1 2 | 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | A Valid values of A: 00000_8-17777_8	Jump to address in program storage specified by A-field; this address is loaded into the Address Register and the Program Counter.	SC = WC = $\overline{LB}$ =	L L X	L L X

NOTES * $\overline{RB}$ is complement of $\overline{LB}$, X = Undefined

AND—data in source register or I/O-bus input is ANDed with content of AUX (R0) register and the result is placed in the destination register or I/O-bus output. Data can be shifted and/or masked, as required.

XOR—data in source register or I/O-bus input is exclusively ORed with contents of AUX (R0) register and the result is placed in the destination register or I/O-bus output. Data can be shifted and/or masked, as required.

XMIT—immediate data field of instruction word replaces data in destination register or I/O-bus output.

XEC—executes instruction at the program address which is formed by replacing the least significant bits of the last address with the sum of:

- Literal (J) field value of instruction plus,
- Value of data in source register or I/O-bus input.

NZT—least significant bits of program address are replaced by literal (J) field of instruction if the source register or I/O-bus is not equal to zero.

JMP—program address is replaced by address field of the instruction word.

Instruction Fields

As shown in Table 1, each instruction contains an operations code (OPCODE) field and from one-to-three operand fields. The operand fields are: Source (S), Destination (D), Rotate/Length (R/L), Literal (J), and Address (A). The OPCODE and operand fields are briefly described in the following paragraphs.

Operations Code Field: The three-bit OPCODE field specifies one of eight classes of 8X300 instructions; octal designations for this field and operands for each instruction class are shown in Table 1.

Source (S) and Destination (D) Fields: The five-bit (S) and (D) fields specify the source and destination of data for the operation defined by the OPCODE field. The AUXiliary (R0) register is an implied second operand for the ADD, AND, and XOR instructions, each of which require two source fields. That is, instructions of the form:

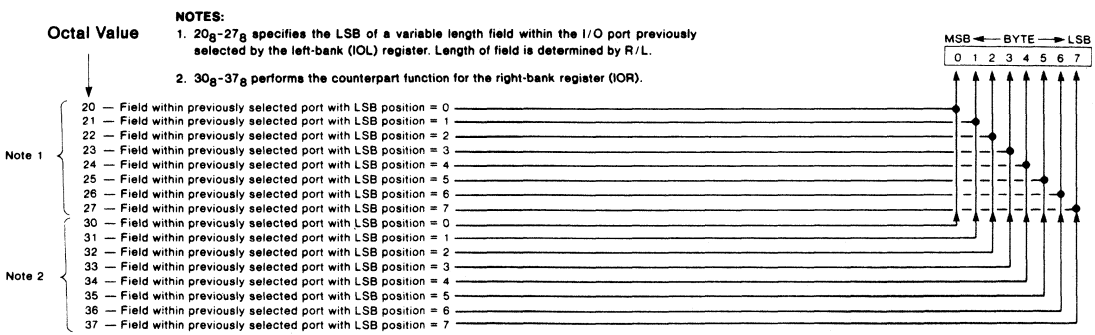
ADD X, Y

imply a third operand, say Z, located in the AUX (R0) register. Thus, the operation for the preceding expression is actually (X + Z), with the result stored in Y. The (S) and/or (D) fields can specify an internal 8X300 register or any one-to-eight bit I/O field; octal values for these registers and Source/Destination field assignments are provided in Table 2.

Table 2. OCTAL ADDRESSES OF 8X300 REGISTERS AND ADDRESS/BIT ASSIGNMENTS OF SOURCE/DESTINATION FIELDS

Octal Value	8X300 Register	Octal Value	8X300 Register
00	Auxiliary (R0)	10	OVF (Overflow Register)—used only as a source
01	R1	11	R11
02	R2	12	Unassigned
03	R3	13	Unassigned
04	R4	14	Unassigned
05	R5	15	Unassigned
06	R6	16	Unassigned
07	*IOL Register—Left Bank I/O Address Register; Used only as destination	17	*IOR Register—Right Bank I/O Address Register; Used only as destination

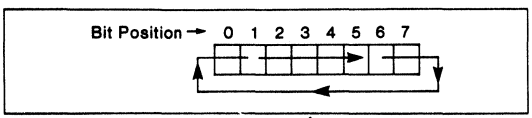
NOTE
*If IOL or IOR is specified as a source of data, the source data is all zeroes.



Rotate (R) and Length (L) Field: The three-bit R/L field performs one of two functions, specifying either the field length (L) or a right-rotate (R). For a given instruction, the specified function depends upon the contents of the source (S) and destination (D) fields.

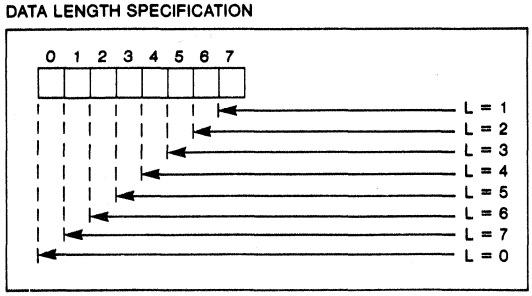
- When an internal register is specified by both the source and destination fields, the (R) field is invoked and it specifies a right-rotate of the data specified in the (S) field—see accompanying diagram. The source-register data (up to eight-bits) is right-rotated within one instruction cycle. (The right-rotate function is implemented on the bus and not in the source register.)

RIGHT-ROTATE FUNCTION



- When either or both of the source and destination fields specify a variable-length I/O data field, the (L) field specifies the length of the I/O data field—see accompanying diagram. If the source field specifies an I/O address (20_8-37_8) and the destination field specifies an internal register (00_8-06_8, 07_8, 11_8, or 17_8), the L-field specifies the length of source data; the source data is formed by right-rotating the I/O bus data according to the source address (Table 2) and then masking result as specified by L-field. If length is less than eight-bits, all remaining bits are set to zero prior to processing data in the ALU. If the source field specifies an internal register (00_8,-06_8, 10_8,or 11_8) and the destination field specifies I/O bus data (20_8-37_8), the L field specifies the length of the destination data. To form the destination data, the ALU output is left-shifted according to the destination address (Table 2) and then masked to the required length—see DATA LENGTH SPECIFICATION. The destination data is merged with data in the I/O latches to finalize the I/O bus data. Hence, a one-to-eight bit destination data field can be inserted into the existing eight-bit I/O port without modifying surrounding bits. If both the source and destination fields specify I/O bus data (20_8-37_8), the L-field specifies the length of both the source and destination data.

DATA LENGTH SPECIFICATION



To form the source data, the I/O bus data is right-rotated according to the source address (Table 2) and then masked to the required length—see preceding DATA LENGTH SPECIFICATION. If length is less than eight-bits, all remaining bits are set to zero before processing in the ALU. To form the destination data, the ALU output is left-shifted according to the destination address (Table 2) and masked to the required length specification. The destination data is then merged into the I/O bus data that was used to obtain the source; thus, if the source and destination addresses are on the same bank, the I/O bus data written to the destination register appears unmodified, except for bits changed during the shift-and-mask operations. If the source and destination addresses refer to different banks, the destination register is changed to contain the contents of the source register in those bit positions not affected by the destination data.

J-Field: The 5-bit or 8-bit (J) field is used to load a literal value (contained in the instruction) into a register, into a variable I/O data field, or to modify the low-order bits of the Program Counter. The bit-length of the (J) field is implied by the (S) field in the XEC, NZT, and XMIT instructions, based on the following considerations.

- When the source (S) field specifies an internal register, the literal value of the J-field is an 8-bit binary number.
- When the source (S) field specifies a variable I/O data field, the literal value of the J-field is a 5-bit binary number.

A-Field: The 13-bit (A) field is an address field which allows the 8X300 to directly address up 8192 locations in Program Storage memory.

INSTRUCTION SEQUENCE CONTROL

Formation of Instruction Address

The Address Register and Program Counter are used to generate addresses for accessing an instruction from program storage. The instruction address is formed in any one of four ways:

- For all except the JMP, XEC, and a "satisfied" NZT instruction, the Program Counter is incremented by one and placed in the Address Register.
- For the JMP instruction, the 13-bit A-field contained in the JMP instruction word replaces the contents of both the Address Register and Program Counter.
- For the XEC instruction, the Address Register is loaded with the high-order bits of the Program Counter modified as follows:
  XEC using I/O Bus Data: low order 5-bits of ALU output replaces counterpart bits in Address Register.
  XEC using Data from Internal Register: low order 8-bits of ALU output replaces counterpart bits in Address Register.
  The Program Counter is not modified for either of the above conditions.
- For a "satisfied" NZT instruction, the low order 5-bits (NZT source is I/O Bus Data) or low order 8/bits (NZT source is an Internal Register) of both the Address Register and Program Counter are loaded with the literal value specified by J-field of the instruction word.

Data Addressing

The source and/or destination addresses of the data to be operated upon are specified as part of the instruction word. As shown in Table 3, source/destination addresses are specified using a five-bit address (00_8 through 37_8). When the most significant octal digit is a 0 or 1, the source and/or destination address is an internal register; if the most significant digit is a 2 or 3, an I/O bus address is indicated—2 specifying a left-bank (LB) address and 3 specifying a right-bank (RB) address. The least significant octal digit (0 through 7) indicates either a specific internal register address or positioning information for the least significant bit when specifying I/O bus data. Referring to Table 1, the AUXiliary register (00) is the implied source of the second argument for the ADD, AND, and XOR operations. IOL (destination address 07_8) and IVR (destination address 17_8) provide a means of routing address information to I/O registers. With IOL or IOR specified as the destination address, the data is placed on the I/O bus during the output phase of the instruction cycle. Simultaneously, a select command (SC) is generated to inform all I/O devices that information on the I/O bus is to be considered as an I/O address. Since IOL and IOR are not harware registers, they should never be specified as a source address.

Control outputs $\overline{LB}$ and $\overline{RB}$ are used to partition I/O bus devices into two fields of 256 addresses. With LB in the active-low state and a source address of 20_8-27_8, the left bank of I/O devices are enabled during the input phase of the instruction cycle. With $\overline{RB}$ in the active-low state and a source address of 30_8-37_8, the right bank of devices are enabled. During the output phase, $\overline{RB}$ is low if the destination address is IOR (17_8) or 30_8-37_8; $\overline{LB}$ is low if the destination address is IOL (07_8) or 20_8-27_8. Each address field ($\overline{LB}$ and $\overline{RB}$) can have a different I/O device selected; thus, two devices can be directly accessed within one instruction cycle.

Table 3. SOURCE/DESTINATION ADDRESSES

SOURCE AND/OR DESTINATION FIELD (OCTAL)	SOURCE/DESTINATION
00	AUXiliary register (R0)
01-06	Working registers R1-R6, respectively
07	IOL Left-bank enable (Destination only)
10	Overflow status—OVF (Source only)
11	Working register R11
17	IOR Right-bank enable (Destination only)
2N (N = 0, 1, 2, 3 4, 5, 6, or 7)	If a source, I/O data is right-rotated (7 − N) bits and then masked as specified by the L-field. $\overline{LB}$ = low and $\overline{RB}$ = high generated during input phase. If a destination, I/O data is left-shift (7 − N) bits and merged (specified by L-field) with data contained in the I/O latches. $\overline{LB}$ = low and $\overline{RB}$ = high generated during output phase.
3N (N = 0, 1, 2, 3, 4, 5, 6, or 7)	If a source, I/O data is right-rotated (7 − N) bits and then masked as specified by the L-field. $\overline{LB}$ = high and $\overline{RB}$ = low generated during input phase. If a destination, I/O data is left-shifted (7 − N) bits and merged (specified by L-field) with data contained in the I/O latches. $\overline{LB}$ = high and $\overline{RB}$ = low generated during output phase.

DESIGN PARAMETERS

Hardware design of an 8X300-based system largely consists of the following operations:

- **Selecting and interfacing a Program Storage device—ROM, PROM, etc. (Pins 2 through 9 and 45 through 49 for 13-bit address interface; Pins 13 through 28 for 16-bit instruction interface.)**
- **Selecting and interfacing Input/Output devices—RAM, Multiplexers, I/O Ports, and other eight-bit addressable I/O devices. (Pins 33 through 36 and pins 38 through 41 for eight-bit I/O interface.)**
- **Choosing and implementing System Clock—Capacitor-Controlled, Crystal-Controlled, or Externally-Driven. (Pins 10 and 11 for System Clock interface.)**
- **Selection of 5-volt power supply and off-chip series-pass transistor.**
- **External logic, as required, to meet the control requirements of a particular application.**

All information required for easy implementation of these design requirements is provided under the following captions.

- **DC Characteristics**
- **AC Characteristics**
- **Timing Considerations**
- **Clock Considerations**
- **HALT/RESET Logic**
- **Voltage Regulator**

DC CHARACTERISTICS (Commercial Part) $4.75V \leq V_{CC} \leq 5.25V$, $0°C \leq T_A \leq 70°C$

PARAMETER		TEST CONDITIONS	LIMITS			UNIT	COMMENTS
			Min	Typ	Max		
V_{CC}	Supply voltage		475	5.0	5.25	V	5V ± 5%; pin 37 only
V_{IH}	High level input voltage		0.6 2.0		2.0	V	X1 and X2 All other pins
V_{IL}	Low level input voltage				0.4 0.8	V	X1 and X2 All other pins
V_{OH}	High level output voltage	V_{CC} = min; I_{OH} = −3mA	2.4	3.0		V	
V_{OL}	Low level output voltage	V_{CC} = min; I_{OL} = 6mA V_{CC} = min; I_{OL} = 16mA		0.39 0.39	0.55 0.55	V	A0 through A12 All other outputs
V_{CR}	Regulator voltage	V_{CC} = 5V		3.1		V	From series-pass transistor
V_{IC}	Input clamp voltage	V_{CC} = min; I_{IN} = −10mA			−1.5	V	Crystal inputs X1 and X2 do not have internal clamp diodes.
I_{IH}	High-level input current	V_{CC} = max; V_{IH} = 0.6V V_{IH} = 4.5V		1	3.0 50	mA μA	X1 and X2 All other pins
I_{IL}	Low-level input current	V_{CC} = max; V_{IL} = 0.4V		 −0.13 −0.67 −0.23	−3 −0.2 −1.6 −0.4	mA	X1 and X2 $\overline{IV0}$-$\overline{IV7}$ I0-I15 $\overline{HALT}$ and $\overline{RESET}$
I_{OS}	Short circuit output current	V_{CC} = max; V_{CR} = V_{CRH} (Note: At any time, no more than one output should be connected to ground.)	−30		−140	mA	All output pins
I_{CC}	Supply current	V_{CC} = max; V_{CR} = V_{CRH}			160	mA	
I_{REG}	Regulator control	V_{CC} = 5.0V	−14		−21	mA	
I_{CR}	Regulator current	V_{CC} = max			230	mA	70°C
					265	mA	25°C
					290	mA	0°C

NOTES:
1. Operating temperature ranges are guaranteed after thermal equilibrium has been reached.
2. All voltages measured with respect to ground terminal.

AC CHARACTERISTICS (Commercial Part)

CONDITIONS: V_{CC} = 5V (±5%), V_{IN} = 0V or 3V, $0°C \leq T_A \leq 70°C$
LOADING: (See test circuits)

PARAMETER (NOTE 1)		LIMITS (INSTRUCTION CYCLE TIME = 250 ns)			LIMITS (INSTRUCTION CYCLE TIME > 250 ns)			UNITS	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
T_{PC}	Processor cycle time	250			250			ns	
T_{CP}	X1 clock period	125			125			ns	
T_{CH}	X1 clock high time	62			62			ns	
T_{CL}	X1 clock low time	62			62			ns	
T_{MCH}	MCLK high delay	31	42	52	31	42	52	ns	
T_{MCL}	MCLK low delay	31	42	52	31	42	52	ns	
T_W	MCLK pulse width	55	62	69	$T_{4Q}-7$	T_{4Q}		ns	Note 2
T_{AS}	X1 falling edge to address stable	50	63	80	50	63	80	ns	Note 7

AC CHARACTERISTICS (Commercial Part) (Continued)

CONDITIONS: V_{CC} = 5V (±5%), V_{IN} = 0V or 3V, 0°C ≤ T_A ≤ 70°C
LOADING: (See test circuits)

PARAMETER (NOTE 1)		LIMITS (INSTRUCTION CYCLE TIME = 250 ns)			LIMITS (INSTRUCTION CYCLE TIME > 250 ns)			UNITS	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
T_{MAS}	MCLK falling edge to address stable	130	143	160	$T_{1Q}+T_{2Q}$ +5	$T_{1Q}+T_{2Q}$ +18	$T_{1Q}+T_{2Q}$ +35	ns	Notes 2, 3, & 7
T_{IA}	Instruction to address			170			T_{2Q}+108	ns	Notes 2, 3, & 8
T_{IVA}	Input data to address			105			105	ns	Notes 3 & 9
T_{IS}	Instruction set-up time (X1 rising edge)	−7			−7			ns	Note 10
T_{MIS}	MCLK falling edge to instruction stable			20			T_{1Q}−42	ns	Notes 2, 4, & 10
T_{IH}	Instruction hold time (X1 rising edge)	45			45			ns	Note 11
T_{MIH}	Instruction hold time) (MCLK falling edge)	60			T_{1Q}−2			ns	Notes 2 & 11
T_{WH}	X1 falling edge to SC/WC rising edge	40	49	58	40	49	58	ns	
T_{MWH}	MCLK falling edge to SC/WC rising edge	125	130	135	$T_{1Q}+T_{2Q}$	$T_{1Q}+T_{2Q}$ +5	$T_{1Q}+T_{2Q}$ +10	ns ns	Note 2
T_{WL}	X1 falling edge to SC/WC falling edge	40	49	58	40	49	58	ns	
T_{MWL}	MCLK falling edge to SC/WC falling edge	5	7	15	5	7	15	ns	
T_{IBS}	X1 falling edge to $\overline{LB}/\overline{RB}$ (Input phase)	48	60	70	48	60	70	ns	
T_{MIBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Input phase)	7	17	25	7	17	25	ns	
T_{IIBS}	Instruction to $\overline{LB}/\overline{RB}$ (Input phase)		27	35		27	35	ns	
T_{OBS}	X1 falling edge to $\overline{LB}/\overline{RB}$ (Output phase)	48	60	70	48	60	70	ns	
T_{MOBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Output phase)	132	137	147	$T_{1Q}+T_{2Q}$ +7	$T_{1Q}+T_{2Q}$ +12	$T_{1Q}+T_{2Q}$ +22	ns	Note 2
T_{IDS}	Input data set-up time (X1 falling edge)	25	16		25	16		ns	
T_{MIDS}	MCLK falling edge to input data stable		65	55		$T_{1Q}+T_{2Q}$ −60	$T_{1Q}+T_{2Q}$ −70	ns	Notes 2 & 5
T_{IDH}	Input data hold time (X1 falling edge)	40	30		40	30		ns	
T_{MDIH}	Input data hold time (MCLK falling edge)	125	112		$T_{1Q}+T_{2Q}$	$T_{1Q}+T_{2Q}$ −13		ns	Note 2
T_{ODH}	Output data hold time (X1 falling edge)	55	65	75	55	65	75	ns	
T_{MODH}	Output data hold time (MCLK falling edge)	11	20	25	11	20	25	ns	
T_{ODS}	Output data stable (X1 falling edge)	74	84	94	74	84	94	ns	Notes 12, 14, & 15
T_{MODS}	Output data stable (MCLK falling edge)	150	160	170	$T_{1Q}+T_{2Q}$ +25	$T_{1Q}+T_{2Q}$ +35	$T_{1Q}+T_{2Q}$ +45	ns	Notes 2, 12, 14, & 15

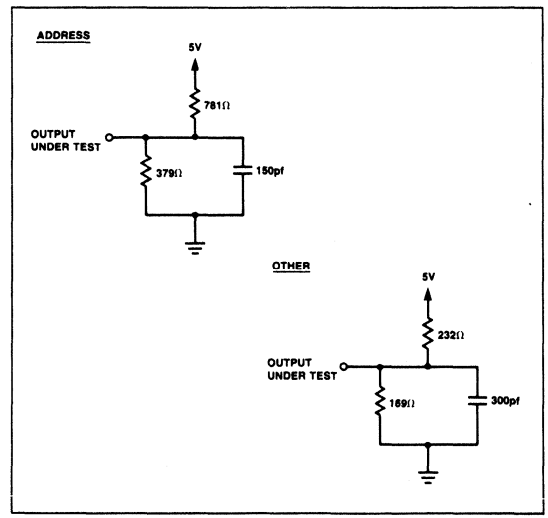
AC CHARACTERISTICS (Commercial Part) (Continued)

CONDITIONS: V_{CC} = 5V (±5%), V_{IN} = 0V or 3V, 0°C ≤ T_A ≤ 70°C
LOADING: (See test circuits)

PARAMETER (NOTE 1)		LIMITS (INSTRUCTION CYCLE TIME = 250 ns)			LIMITS (INSTRUCTION CYCLE TIME > 250 ns)			UNITS	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
T_{DD}	Input data to output data	104	120	136	104	120	136	ns	Notes 13 & 15
T_{HS}	$\overline{HALT}$ set-up time (X1 rising edge)	0			0			ns	
T_{MHS}	MCLK falling edge to $\overline{HALT}$ falling edge			18			$T_{1Q}-44$	ns	Notes 2 & 6
T_{HH}	$\overline{HALT}$ hold time (X1 rising edge)	32			32			ns	
T_{MHH}	$\overline{HALT}$ hold time (MCLK falling edge)	50			$T_{1Q}-12$			ns	Note 2
T_{ACC}	Program storage access time			80				ns	
T_{IO}	I/O port output enable time ($\overline{LB}/\overline{RB}$ to valid IV data input)			30				ns	

NOTES:
1. X1 and X2 inputs are driven by an external pulse generator with an amplitude of 1.5 volts; all timing parameters are measured at this voltage level.
2. Respectively, T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q} represent time intervals for the first, second, third, and fourth quarter cycles.
3. Capacitive loading for the address bus is 150 picofarads.
4. Same as TIS but referenced to falling edge of MCLK.
5. Same as TIDS but referenced to falling edge of MCLK.
6. Same as THS but referenced to falling edge of MCLK.
7. TAS is obtained by forcing a valid instruction and an I/O bus input to occur earlier than the specified minimum set-up time; the TAS parameter then represents the earliest time that the address bus is valid.
8. TIA is obtained by forcing a valid instruction input to occur earlier than the minimum set-up time.
9. TIVA is obtained by forcing a valid I/O bus input to just meet the minimum set-up time.
10. TMIS represents the set-up time required by internal latches of the 8X300. In system applications, the instruction input may have to be valid before the worst-case set-up time in order for the system to respond with a valid I/O bus input that meets the I/O bus input set-up time (TIDS and TMIDS).
11. TIH represents the hold time required by internal latches of the 8X300. To generate proper LB/RB signals, the instruction must be held valid until the address bus changes.
12. TODS is obtained by forcing a valid I/O bus input to occur earlier than the I/O bus input set-up time (TIDS); this timing parameter represents the earliest time that the I/O output data can be valid.
13. TDD is obtained by forcing a valid I/O bus input to just meet the minimum I/O bus input set-up time; thus timing parameter represents the latest time that the I/O output data can be valid.
14. The minimum figure for these parameters represents the earliest time that I/O bus output drivers of the 8X300 will turn on.
15. For TIDS ≥ 35 ns, TODS or TMODS should be used to determine when the output data is stable.

TEST CIRCUITS



DC CHARACTERISTICS (Military Part)

S8X300-1 $-40°C \leq TC \leq 100°C$ $V_{CC} = 5V \pm 5\%$
S8X300-2 $-20°C \leq TC \leq 100°C$ $V_{CC} = 5V \pm 10\%$

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IH}	High level input voltage X1, X2 All others		 0.6 2.0			 V V
V_{IL}	Low level input voltage X1, X2 All others				 0.4 0.8	 V V
V_{IC}	Input clamp voltage (Notes 1 & 5)	V_{CC} = min $I_I = -10mA$			−1.5	V
I_{IH}	High level input current X1, X2 All others	 V_{CC} = max V_{IH} = 0.6V V_{CC} = max V_{IH} = 4.5V			 3.0 0.05	 mA
I_{IL}	Low level input current X1, X2	 V_{CC} = max V_{IL} = 0.4V			 −3.0	 mA
	$\overline{IV0}$-$\overline{IV7}$	V_{CC} = max V_{IL} = 0.4V			−0.3	mA
	I0-I15	V_{CC} = max V_{IL} = 0.4V			−1.6	mA
	$\overline{HALT}$, $\overline{RESET}$	V_{CC} = max V_{IL} = 0.4V			−0.4	mA
V_{OL}	Low level output voltage A0-A12 All others	 V_{CC} = min I_L = 4.25mA V_{CC} = min I_{OL} = 16mA			 0.55 0.55	 V V
V_{OH}	High level output voltage	V_{CC} = min $I_{OH} = -3mA$	2.4			V
I_{OS}	Short circuit output current (Note 2)	V_{CC} = max	−30		−140	mA
I_{CC}	Supply current (Note 4)	V_{CC} = max			160	mA
I_{REG}	Regulator control	V_{CC} = 5.0V	−14		−21	mA
I_{CR}	Regulator current	V_{CC} = max			285	mA
I_{CR}	Regulator current	$TC \geq 25°C$ V_{CC} = max			330	mA
V_{CR}	Regulator voltage	$TC < 25°C$ (Note 3)		3.1		V

NOTES:
1. Crystal inputs X1 and X2 do not have clamp diodes.
2. Only one output may be grounded at a time.
3. From series-passed transistor under the following conditions:
   VCC = Max, HALT = RESET = ADDRESS = IVX = 0.0V, all other pins open.
4. Pin 37 only.
5. Test each input one at a time.
6. All voltages are with respect to ground terminal.
7. The operating temperature ranges are guaranteed after thermal equilibrium has been reached.
8. Storage temperature −65°C to +150°C.

AC CHARACTERISTICS (Military Part)

CONDITIONS: S8X300-1—V_{CC} = 5V (±5%) -40°C ≤ T_C ≤ 100°C
S8X300-2—V_{CC} = 5V (±10%) -20°C ≤ T_C ≤ 100°C

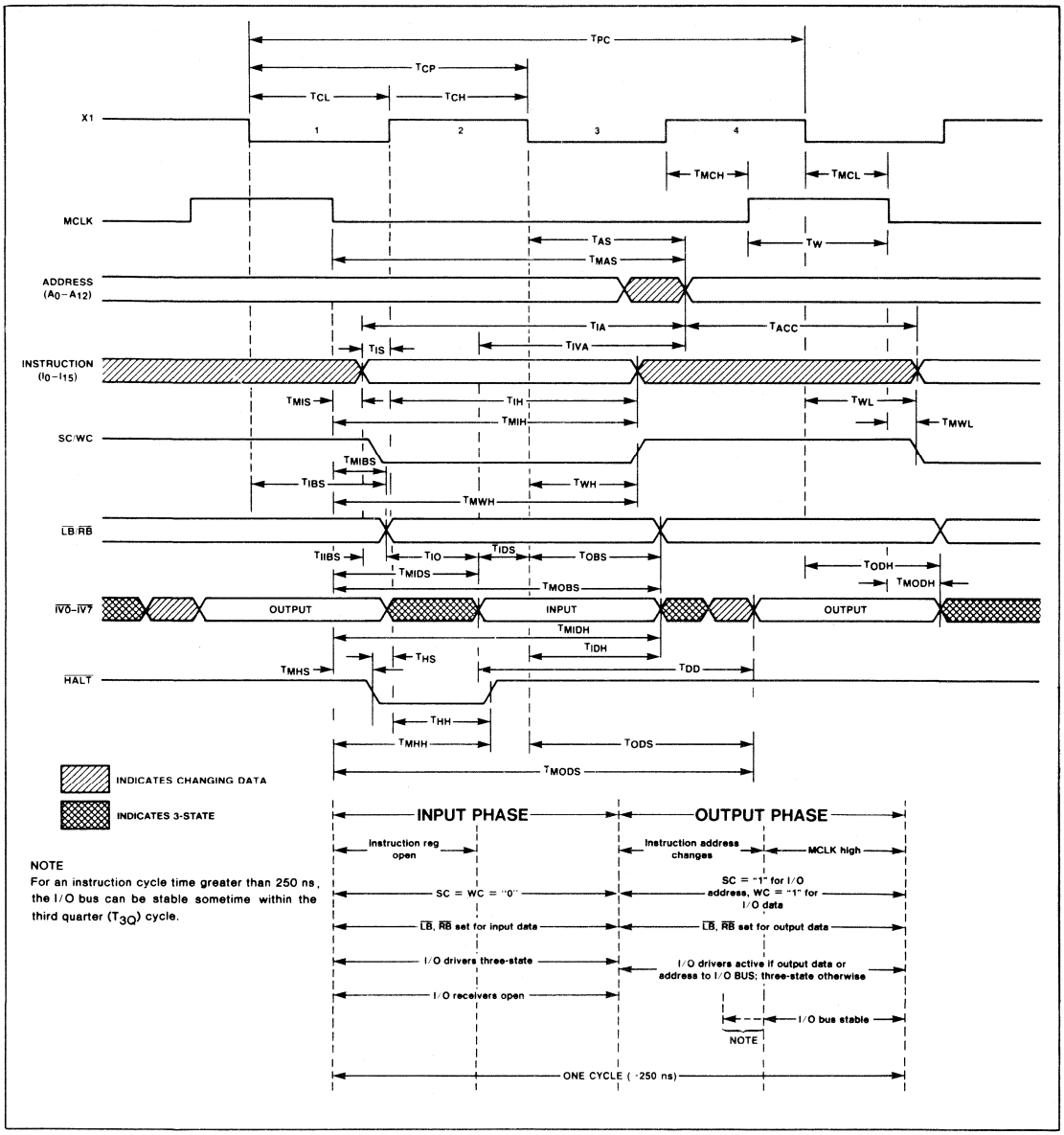
PARAMETER		TEST CONDITIONS (NOTES 1 & 2)	LIMITS			UNIT
			Min	Typ	Max	
Clock:						
T_{PC}	Processor cycle time		300			ns
T_{CP}	X1 clock period		150			ns
T_{CH}	X1 clock high time		62			ns
T_{CL}	X1 clock low time		62			ns
Controls:						
T_{HS}	$\overline{HALT}$ set-up time (X1 rising edge)		0			ns
T_{HH}	$\overline{HALT}$ hold time (X1 rising edge)		50			ns
Instructions:						
T_{AS}	X1 falling edge to address stable	CL = 100pF	35		92	ns
T_{IS}	Instruction set-up time (X1 rising edge)		0			ns
T_{IH}	Instruction hold time (X1 rising edge)		50			ns
T_{MCH}	MCLK high delay	X1 = 2.0V	20		55	ns
T_{MCL}	MCLK low delay	X1 = 2.0V	20		55	ns
T_{WH}	X1 falling edge to SC/WC rising edge				80	ns
T_{WL}	X1 falling edge to SC/WC falling edge				80	ns
T_{IIBS}	Instruction to $\overline{LB}/\overline{RB}$ (input phase)				52	ns
T_{IBS}	X1 falling edge to $\overline{LB}/\overline{RB}$ (input phase)		24			ns
T_{OBS}	X1 falling edge to $\overline{LB}/\overline{RB}$ (output phase)				90	ns
T_{IDS}	Input data set-up time (X1 falling edge)		36			ns
T_{IDH}	Input data hold time (X1 falling edge)		50			ns
T_{ODS}	Output data stable (X1 falling edge)				125	ns
T_{ODH}	Output data hold time (X1 falling edge)		35		85	ns
T_{ACC}	Instruction access time	Provided by worst case timing	80			ns
T_{IO}	Data I/O access time	Provided by worst case timing	40			ns

NOTES:
1. Operating temperature ranges are guaranteed after thermal equilibrium has been reached.
2. Unless otherwise noted CL = 300 pF, VIN = 3V.

TIMING CONSIDERATIONS (Commercial Part)

As shown in the "AC CHARACTERISTICS" table for this part, the minimum instruction cycle time is 25 ns, whereas, the maximum is determined by the on-chip oscillator frequency and can be any value the user chooses. With an instruction cycle time of 250 ns, the part can be characterized in terms of absolute values; these are shown in the first "LIMITS" column of the table. When the instruction cycle time is greater than 250 ns, certain parameters are cycle-time dependent; thus, these parameters are specified in terms of the four quarter cycles (T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q}) that make up one instruction cycle—see 8X300 TIMING DIAGRAM. As the time interval for each instruction cycle increases (becomes greater than 250 ns), the delay for all parameters that are cycle-time dependent is likewise increased. In some cases, these delays have a significant impact on timing relationships and other areas of systems design; subsequent paragraphs describe these timing parameters and reliable methods of calculation.

8X300 TIMING DIAGRAM



NOTE
For an instruction cycle time greater than 250 ns, the I/O bus can be stable sometime within the third quarter (T_{3Q}) cycle.

Timing parameters for the 8X300 are normally measured with reference to X1 or MCLK; those referenced to MCLK are prefaced with an "M" in the mnemonic—TMAS, TMIH, and so on. To determine the timing relationship between a particular signal, say "A" and MCLK, the user should, at all times, use the value specified in the table—DO NOT calculate the value by adding or subtracting two or more parameters that are referenced to X1. When deriving timing relationships between two signals (A to B, etc.) by adding or subtracting the parameter values, the user must consistently use the same parameter reference—MCLK or X1.

System determinants for the instruction cycle time are:

- Propagation delays within the 8X300
- Access time of Program Storage
- Enable time of the I/O port

Normally, the instruction cycle time is constrained by one or more of the following conditions:

Condition 1—Instruction or MCLK to $\overline{LB}/\overline{RB}$ (input phase) plus I/O port access time (TIO) ≤ IV data set-up time (Figure 4a).

Condition 2—Program storage access time (TACC) plus instruction to $\overline{LB}/\overline{RB}$ (input phase) plus I/O port access time (TIO) plus IV data (input phase) to address ≤ instruction time (Figure 4b).

Condition 3—Program storage access time plus instruction to address ≤ instruction cycle time (Figure 4c).

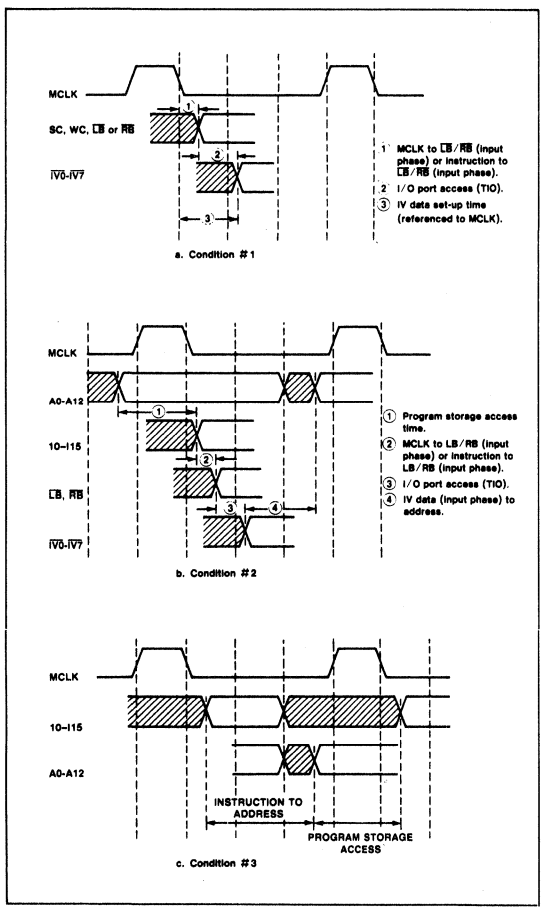


Figure 4. Constraints of 8X300 Instruction Cycle Time

From condition #1 and with an instruction cycle time of 250 ns, the I/O port access time (TIO) can be calculated as follows:

$$TMIBS + TIO \leq TMIDS$$
$$\text{transposing,}\quad TIO \leq TMIDS - TMIBS$$
$$\text{substituting,}\quad TIO \leq 55ns - 25ns$$
$$\text{result,}\quad TIO \leq 30\ ns$$

Using 30 ns for TIO, the constraint imposed by condition #1 can also be used to calculate the minimum cycle time:

$$TMIBS + TIO \leq TMIDS$$
$$\text{thus,}\quad 25ns + 30ns \leq T_{1Q} + T_{2Q} - 70$$
$$25ns + 30ns \leq \tfrac{1}{2}\ \text{cycle} - 70$$

therefore, the worst-case instruction cycle time is 250 ns. With subject parameters referenced to X1, the same calculations are valid:

$$TIBS + TIO + TIDS \leq \tfrac{1}{2}\ \text{cycle}$$
$$\text{thus,}\quad 70ns + 30ns + 25ns \leq \tfrac{1}{2}\ \text{cycle}$$

therefore, the worst-case instruction cycle time is again 250 ns. From condition #2 and with an instruction cycle time of 250 ns, the program storage access time can be calculated:

$$TACC + TIIBS + TIO + TIVA \leq 250ns$$
$$\text{transposing,}\quad TACC \leq 250ns - TIIBS - TIO - TIVA$$
$$\text{substituting,}\quad TACC \leq 250ns - 35ns - 30ns - 105ns$$
$$\text{thus,}\quad TACC \leq 80ns$$

hence, for an instruction cycle time of 250 ns, a program storage access time of 80 ns is implied. The constraint imposed by condition #3 can be used to verify the maximum program storage access time:

$$TIA + TACC \leq \text{Instruction Cycle}$$
$$\text{thus,}\quad TACC \leq 250ns - 170ns$$
$$\text{and,}\quad TACC \leq 80ns,$$

confirming that a program storage access time of 80 ns is satisfactory.

For an instruction cycle time of 250 ns and a program storage access time of 80 ns (Condition #2/Figure 4b), the instruction should be valid 10 ns before the falling edge of MCLK. This relationship can be derived by the following equation:

$$250ns - TMAS - TACC$$
$$= 250ns - 160ns - 80ns$$
$$= 10ns$$

It is important to note that, during the input phase, the beginning of a valid $\overline{LB}/\overline{RB}$ signal is determined by either the instruction to $\overline{LB}/\overline{RB}$ delay (TIIBS) or the delay from the falling edge of MCLK to $\overline{LB}/\overline{RB}$ (TMIBS). Assuming the instruction is valid 10 ns before the falling edge of MCLK and adding the instruction-to-LB/RB delay (TIIBS) = 30ns), the $\overline{LB}/\overline{RB}$ signal will be valid 25 ns after the falling edge of MCLK. With a fast program storage memory and with a valid instruction more than 10 ns before the falling edge of MCLK—the $\overline{LB}/\overline{RB}$ signal will, due to the TMIBS delay, still be valid 25 ns after the falling edge of MCLK. Using a worst-case instruction cycle time of 250 ns, the user cannot gain a speed advantage by selecting a memory with faster access time. Under the same conditions, a speed advantage cannot be obtained by using an I/O port with fast access time (TIO) because the address bus will be stable 80 ns (TAS) after the beginning of the third quarter cycle—no matter how early the IV data input is valid.

Internal Timing and Timing Relationships

All timing and timing-control signals of the 8X300 are generated by the oscillator and sequencer shown in Figure 5. The sequencer outputs direct and control all of the timing parameters specified in the TIMING DIAGRAM. Observe that each input quarter cycle bears a fixed relationship to X1 via the propagation delay.

General and interactive timing relationships pertaining to I/O signals of the 8X300 are shown in Figure 6. Example—in the input phase, the switching point of the $\overline{LB}/\overline{RB}$ signal is caused by the worst-case delay from the instruction to $\overline{LB}/\overline{RB}$ or from the beginning of the first internal quarter cycle to $\overline{LB}/\overline{RB}$; the two arrows pointing to the $\overline{LB}/\overline{RB}$ transition indicate this "either/or" dependency. This information coupled with tabular values and the TIMING DIAGRAM provides the user with the wherewithal to calculate any and all system timing parameters.

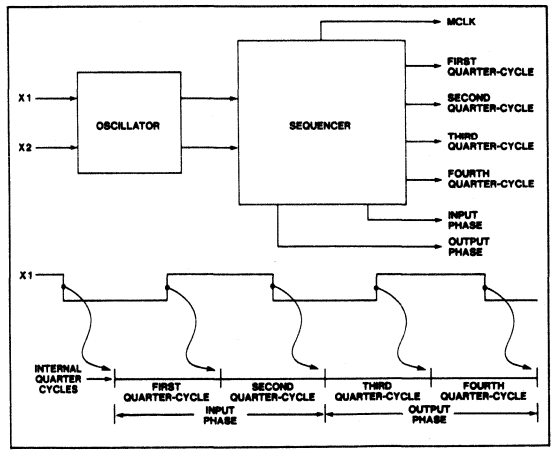


Figure 5. Timing and Timing Control Signals of the 8X300

CLOCK CONSIDERATIONS

The on-chip oscillator and timing-generation circuits of the 8X300 can be controlled by any one of the following methods:

Capacitor: if timing is not critical
Crystal: if precise timing is required
External Drive: if application requires that the 8X300 be synchronized with system clock

Capacitor Timing: A non-polarized ceramic or mica capacitor with a working voltage equal to or greater than 25-volts is recommended. The lead lengths of capacitor should be approximately the same and as short as possible; also, the timing circuits should not be in close proximity to external sources of noise. For various capacitor (C_X) values, the cycle time can be approximated as:

C_X (In pF)	APPROXIMATE CYCLE TIME
100	300 ns
200	500 ns
500	1.1 μs
1000	2.0 μs

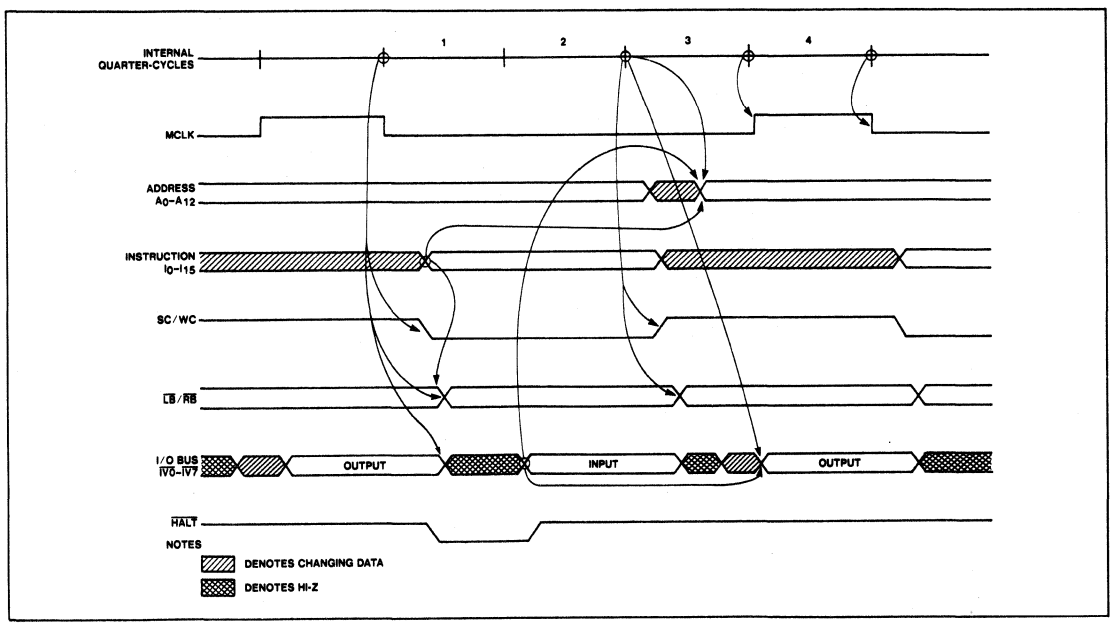


Figure 6. Timing Relationships of 8X300 I/O Signals

Crystal Timing: When a crystal is used, the on-chip oscillator operates at the resonant frequency (f_O) of the crystal; the series-resonant quartz crystal connects to the 8X300 via pins 10 (X1) and 11 (X2). The lead lengths of the crystal should be approximately equal and as short as possible; also, the timing circuits should not be in close proximity to external sources of noise. The crystal should be hermetically sealed (HC type can) and have the following electrical characteristics:
Type: Fundamental mode, series resonant
Impedance at Fundamental: 35-ohms maximum
Impedance at Harmonics and Spurs: 50-ohms minimum

The resonant frequency (f_O) of the crystal is related to the desired cycle time (T) by the equation $f_O = 2/T$; for a cycle time of 250 ns, f_O = 8MHz.

Using an External Clock: The 8X300 can be synchronized with an external clock by simply connecting appropriate drive circuits to the X1/X2 inputs. Figure 7 shows how the on-chip oscillator can be driven from the complementary outputs of a pulse generator. In applications where the microcontroller must be driven from a master clock, the X1/X2 lines can be interfaced to TTL logic as shown in Figure 8.

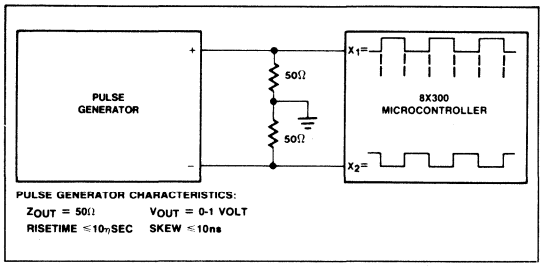


Figure 7. Clocking with a Pulse Generator

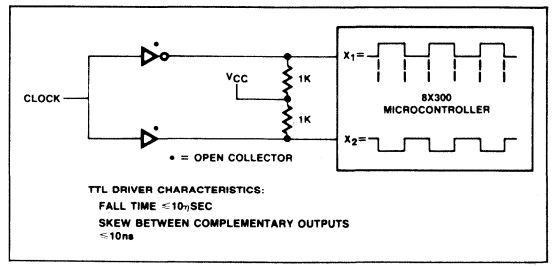


Figure 8. Clocking with TTL

RESET Logic

The $\overline{\text{RESET}}$ line (pin 43) can be driven from a high (inactive) state to a low (active) state at any time with respect to the system clock, that is, the reset function is asynchronous. To ensure proper operation, the $\overline{\text{RESET}}$ line should be held low (active) for one full instruction time. When the line is driven from a high state to an active-low state, several events occur—the precise instant of occurrence is basically a function of the propagation delay for that particular event. As shown in the acompanying $\overline{\text{RESET}}$ timing diagram, these events are:

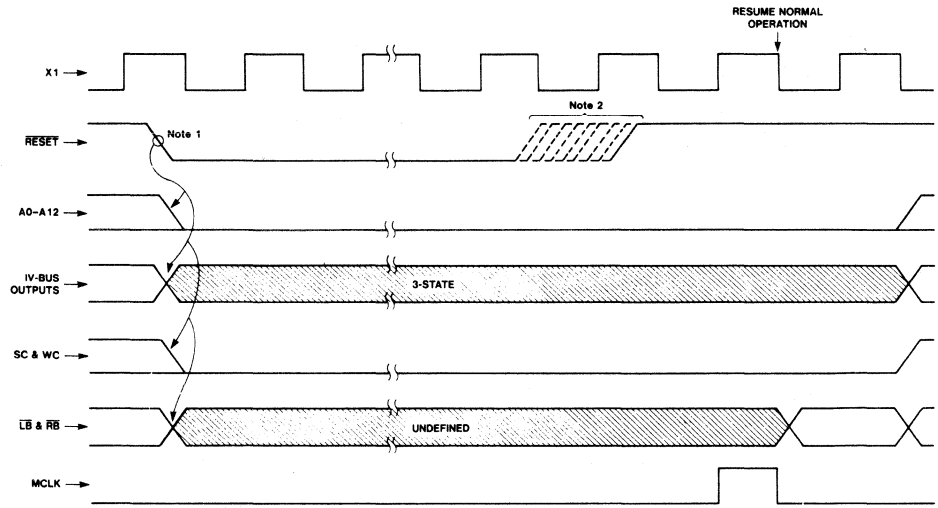
- **The Program Counter and Address Register are set to an all-zero configuration and remain in that state as long as the $\overline{\text{RESET}}$ line is low. Other than PC and AR, reset does not affect other internal registers.**
- **The input/output (IV) bus goes three-state and remains in that mode as long as the $\overline{\text{RESET}}$ line is low.**
- **The Select Command and Write Command signals are driven low and remain inactive as long as the $\overline{\text{RESET}}$ line is low.**
- **The Left Bank/Right Bank signals are undefined for the period in which the $\overline{\text{RESET}}$ line is low.**

During the time $\overline{\text{RESET}}$ is active-low, MCLK is inhibited; moreover, if the $\overline{\text{RESET}}$ line is driven low during the last two quarter cycles, MCLK can be shortened for that particular machine cycle. When $\overline{\text{RESET}}$ line is driven high (inactive)—one-quarter to one full instruction cycle later—MCLK appears just before normal operation is resumed. The $\overline{\text{RESET}}$/MCLK relationship is clearly shown by "B" in the timing diagram. As long as the $\overline{\text{RESET}}$ line is active-low, the $\overline{\text{HALT}}$ signal (described next) is not sampled by internal logic of the 8X300.

HALT Logic

The $\overline{\text{HALT}}$ signal is sampled via internal chip logic at the end of the first internal quarter of each instruction cycle. If, when sampled, the HALT signal is active-low, a halt is immediately executed and the current instruction cycle is terminated; however, the halt cycle does not inhibit MCLK nor does it affect any internal registers of the 8X300. As long as the $\overline{\text{HALT}}$ line is active-low, the SC and WC lines are low (inactive) and the input/output (IV) bus remains in the three-state mode of operation. The halt cycle continues until, when again sampled, the $\overline{\text{HALT}}$ line is found to be high; at this time, normal operation is resumed. Timing for the halt signal is shown in the accompanying diagram.

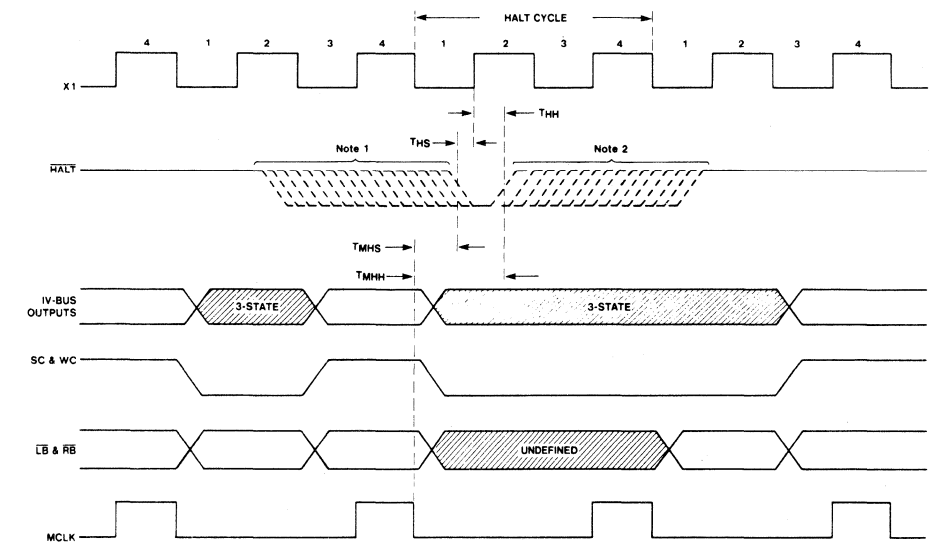
$\overline{\text{RESET}}$ TIMING DIAGRAM



NOTES

1. A High to Low transition of the $\overline{\text{RESET}}$ signal will force the Address Bus to an all-zero configuration.
2. The $\overline{\text{RESET}}$ signal can switch from Low to High at any point within this time interval and, in all cases, MCLK will occur at least one-quarter cycle time later as shown.

$\overline{\text{HALT}}$ TIMING DIAGRAM



NOTES

1. The $\overline{\text{HALT}}$ signal can switch from High to Low at any time during this interval.
2. The $\overline{\text{HALT}}$ signal can switch from Low to High at any time during this interval.

Timing Descriptions:

T_{HS}—set-up time from $\overline{\text{HALT}}$ to X1 (independent of instruction cycle time)

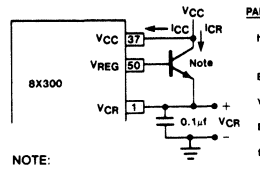
T_{HH}—hold time from X1 to $\overline{\text{HALT}}$ (independent of instruction cycle time)

T_{MHS}—set-up time from MCLK to $\overline{\text{HALT}}$ (dependent upon instruction cycle time)

T_{MHH}—hold time from MCLK to $\overline{\text{HALT}}$ (dependent upon instruction cycle time)

VOLTAGE REGULATOR

All internal logic of the 8X300 is powered by an on-chip voltage regulator that requires an external series-pass transistor. Electrical specifications for the off-chip power transistor and a typical hook-up are shown in the accompanying diagram. To minimize lead inductance, the transistor should be as close as possible to the 8X300 package and the emitter should be ac-grounded via a 0.1-microfarad ceramic capacitor.



PARAMETER	CONDITIONS	LIMITS
h_{fe}	V_{CE} = 2V; 100mA < I_C < 500mA	>50
BE_{ON}	V_{CE} = 5V; I_C = 500mA	<1V
V_{CESAT}	I_C = 500mA; I_B = 50mA	<0.5V
BV_{CEO}		>15V
f_t		>30MHz

NOTE:
Typical approved parts—2N5320, 2N5337

MICROCONTROLLER

Originally published by Signetics January 1984

FEATURES

- **Fetch, Decode, and Execute a 16-bit Instruction in a minimum of 200 nanoseconds (one machine cycle)**
- **Bit-oriented instruction set (addressable single-or-multiple bit subfields)**
- **Separate buses for instruction, instruction address and Three-State I/O**
- **Thirteen 8-bit general-purpose working registers**
- **Source/destination architecture**
- **Bipolar low-power Schottky technology/TTL inputs and outputs**
- **On-chip oscillator and timing generation**
- **Single +5V supply**
- **0.9-in. 50-pin DIP**

PRODUCT DESCRIPTION

The Signetics 8X305 MicroController (Figure 1) is a high-speed bipolar microprocessor implemented with low-power Schottky technology. In a single chip, the 8X305 combines speed, flexibility, and a bit-oriented instruction set. These features and other basic characteristics of the chip combine to provide cost-effective solutions for a broad range of applications. The 8X305 is particularly useful in systems that require high-speed bit manipulations — sophisticated controllers, data communications, very fast interface control, and other applications of a similar nature.

The 8X305 can fetch, decode, and execute a 16-bit instruction word in a minimum of 200 nanoseconds. Within one instruction cycle, the 8-bit data-processing path can be programmed to rotate, mask, shift, and/or merge single or multiple bit subfields and, in addition, perform an ALU operation; in the same instruction, an external data field can be input, processed, and output to a specified destination — likewise, single or multiple bit data fields can be internally moved from a given source to a given destination. To summarize, fixed or variable-length data fields can be fetched, processed, operated on by the ALU, and moved to a different location — all in a timeframe of 200 nanoseconds. To interface with I/O and program memory, the 8X305 uses a 13-bit instruction address bus, a 16-bit instruction bus, an 8-bit bidirectional multiplexed I/O data/address bus and a 5-bit I/O control bus.

A wide selection of I/O devices, interface chips, and special-purpose parts are available for systems use. In most applications, the more powerful 8X305 is functionally interchangeable with its predecessor — the 8X300.

ASSOCIATED DOCUMENTATION

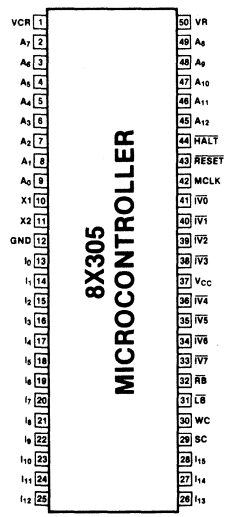
Other documents directly relating to *design* and *applications use* of the 8X305 MicroController are:

- Product Capabilities Manual
- 8X305 Users Manual

These documents and other current literature (Data Sheets, Product Bulletins, Applications Notes, etc.) are available at all Signetics Sales and Service Offices—see rear cover of this data sheet for the office in your locality.

PIN CONFIGURATION

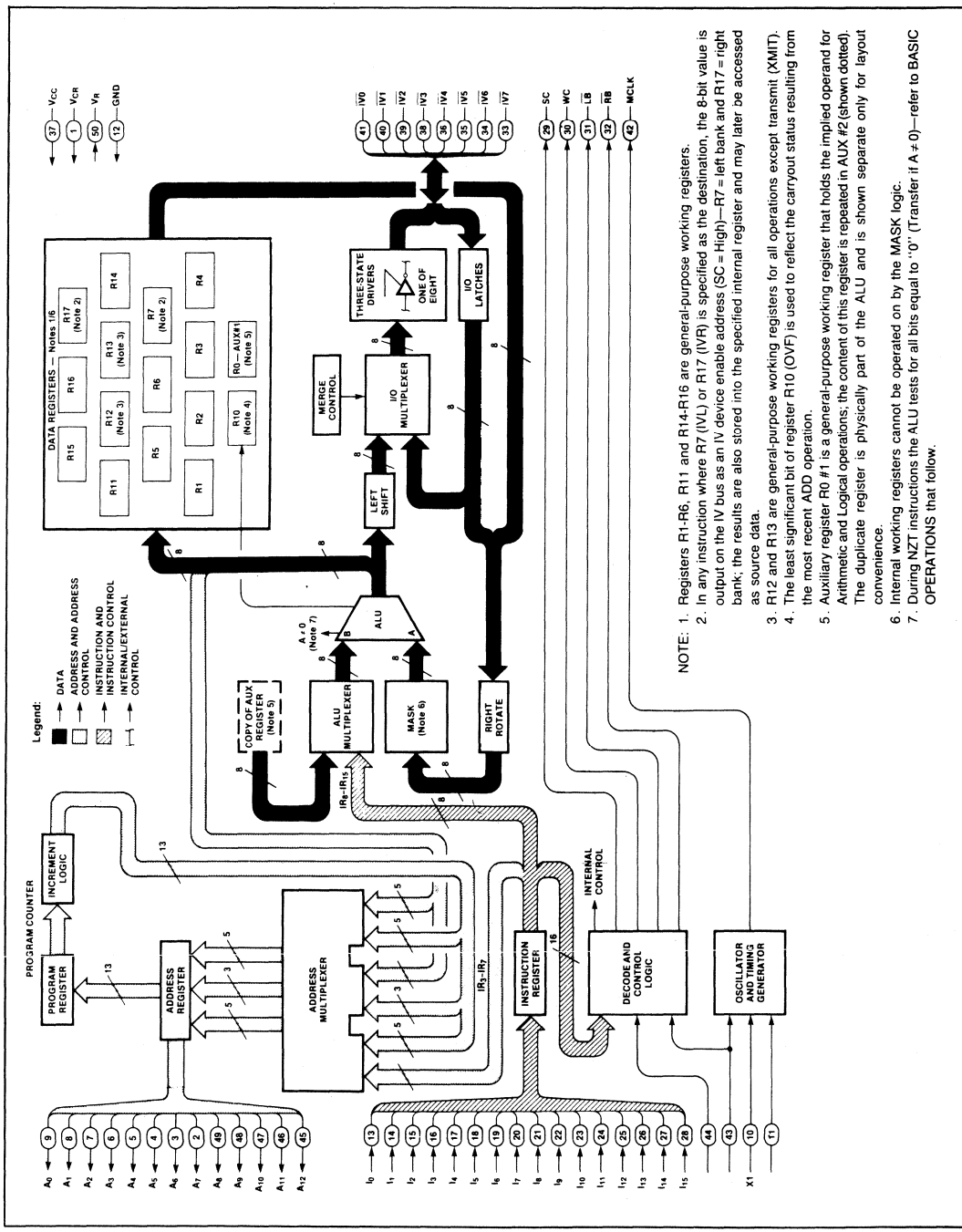
N, I PACKAGE



TOP VIEW

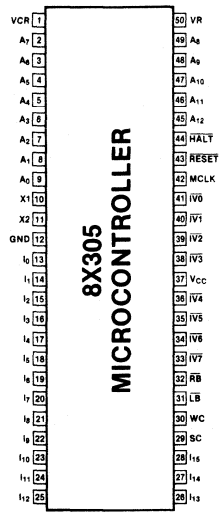
ORDER NUMBERS

8X305N, 8X305I
S8X305I/883B, S8X305I/883C



NOTE: 1. Registers R1-R6, R11 and R14-R16 are general-purpose working registers.
2. In any instruction where R7 (IVL) or R17 (IVR) is specified as the destination, the 8-bit value is output on the IV bus as an IV device enable address (SC = High)—R7 = left bank and R17 = right bank; the results are also stored into the specified internal register and may later be accessed as source data.
3. R12 and R13 are general-purpose working registers for all operations except transmit (XMIT).
4. The least significant bit of register R10 (OVF) is used to reflect the carryout status resulting from the most recent ADD operation.
5. Auxiliary register R0 #1 is a general-purpose working register that holds the implied operand for Arithmetic and Logical operations; the content of this register is repeated in AUX #2 (shown dotted). The duplicate register is physically part of the ALU and is shown separate only for layout convenience.
6. Internal working registers cannot be operated on by the MASK logic.
7. During NZT instructions the ALU tests for all bits equal to "0" (Transfer if A ≠ 0)—refer to BASIC OPERATIONS that follow.

Figure 1. Architecture and Pin Designations for 8X305 MicroController



PIN NO.	IDENTIFIER	FUNCTION
1	VCR	Regulated voltage input from series-pass transistor (2N5320 or equivalent).
2-9, 45-49	$A_0 - A_{12}$	**Program Address Lines:** These active-high outputs permit direct addressing of up to 8192 words of program storage; A_{12} is least significant bit.
10, 11	X1, X2	Timing generator connections for a capacitor, a series resonant crystal, or an external clock source with complementary outputs.
12	GND	Ground.
13-28	$I_0 - I_{15}$	**Instruction Lines:** These active-high input lines receive 16-bit instructions from program storage; I_{15} is least significant bit.
29	SC	**Select Command:** When high (binary 1), an address is being output on pins $\overline{IV0}$ through $\overline{IV7}$.
30	WC	**Write Command:** When high (binary 1), data is being output on pins $\overline{IV0}$ through $\overline{IV7}$.
31	$\overline{LB}$	**Left Bank Control:** When low (binary 0), devices connected to the Left Bank are accessed. (Note. *Typically, the $\overline{LB}$ signal is tied to the $\overline{ME}$ input pin of I/O peripherals).*
32	RB	**Right Bank Control:** When low (binary 0), devices connected to the Right Bank are accessed (Note. *Typically, the $\overline{RB}$ signal is tied to the $\overline{ME}$ input pin of I/O peripherals).*
33-36, 38-41	$\overline{IV0}$-$\overline{IV7}$	**Interface Vector** (Input/Output Bus) — these bidirectional active-low three-state lines communicate data and/or addresses to I/O devices and memory locations. A low voltage level equals a binary "1"; $\overline{IV7}$ is Least Significant Bit.
37	V_{CC}	+5V power supply.
42	MCLK	**Master Clock:** This active-high output signal is used for clocking I/O devices and/or synchronization of external logic.
43	$\overline{RESET}$	When **$\overline{RESET}$** input is low (binary 0), the 8X305 is initialized — sets Program Counter/Address Register to zero and inhibits MCLK. For the period of time $\overline{RESET}$ is low, the Left Bank/Right Bank ($\overline{LB}$/$\overline{RB}$) signals are forced high asynchronously.
44	$\overline{HALT}$	When **$\overline{HALT}$** input is low (binary 0), internal operation of the 8X305 stops at the start of next instruction; MCLK is not inhibited nor is any internal register affected; however, both the Left Bank/Right Bank ($\overline{LB}$/$\overline{RB}$) signals are synchronously driven high during the first quarter of the instruction cycle time and remain high during the time HALT is low.
50	VR	Internally-generated reference output voltage for external series-pass regulator transistor.

Figure 2. Designations and Descriptions for Pins of 8X305 MicroController.

FUNCTIONAL OPERATION

Typical System Configuration

Although the system hookup shown in Figure 3 is of the simplest form, it provides a fundamental look at the 8X305 MicroController and peripheral relationships. As indicated, the 8X305 can directly address up to 8K words of program storage — either ROM or PROM. The user interface ($\overline{IV0}$ through $\overline{IV7}$) is capable of uniquely addressing 256 Input/Output locations and, with additional bank bits ($\overline{LB}$, $\overline{RB}$), this number is expanded to 512 — each bank comprising 256 addressable locations. The addressable locations of each bank can be used in a variety of ways; a simple method of implementation is shown in Figure 3. When $\overline{LB}$ is active low, the left bank is enabled and any one of 256 locations within the RAM memory can be accessed for input/output operations. A similar set of "enable/access" conditions are applicable to the right bank when $\overline{RB}$ is active low.

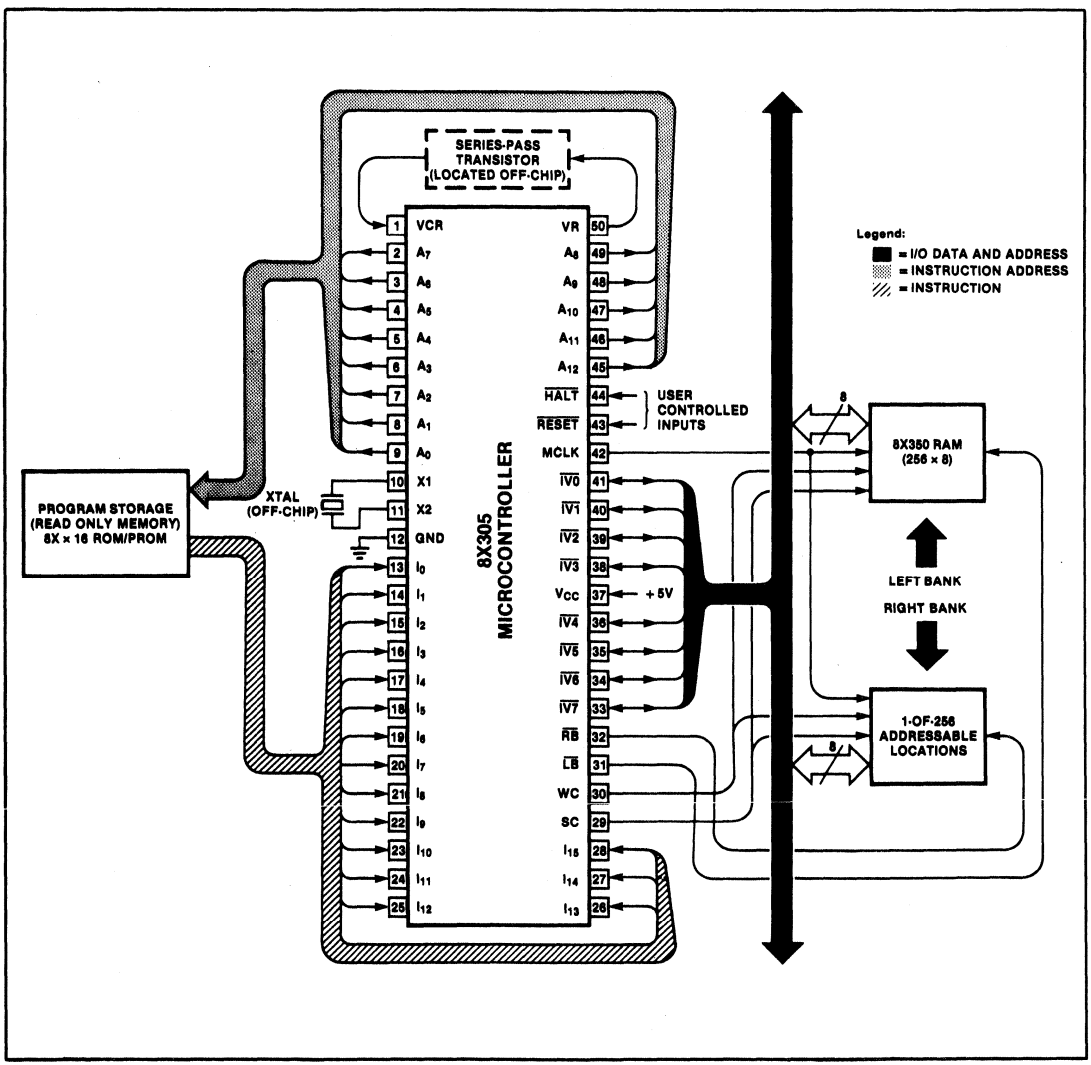


Figure 3. Typical 8X305 System Hookup

BASIC OPERATIONS OF 8X305

Refer to a later discussion of "Instruction Fields" for a detailed examination of all operand fields and subdivisions thereof—"S" (S_0, S_1), "D" (D_0, D_1), "R", "L", "J", and "A".

MOVE OPERATIONS

REGISTER-TO-REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0-R17 as specified by "S" field of instruction.	Right rotate as specified by "R" field of instruction.	No operation.	No operation.	R0-R7, R11-R17 as specified by "D" field of instruction.

REGISTER-TO-$\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0-R17 as specified by "S" field of instruction.	No operation.	No operation.	Shift and merge as specified by "D_0" and "L" fields of instruction.	Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "D_0" and "D_1" fields of instruction.

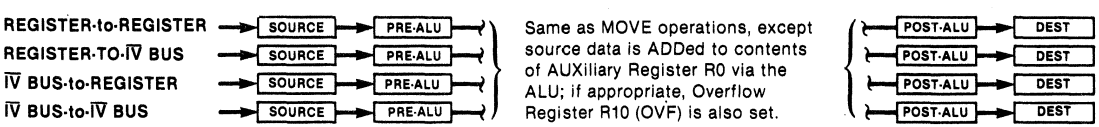
$\overline{IV}$ BUS-TO-REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "S_1" and "S_0" fields of instruction.	Right rotate and mask as specified by "S_0" and "L" fields of instruction.	No operation.	No operation.	R0-R7, R11-R17 as specified by "D" field of instruction.

$\overline{IV}$ BUS-TO-$\overline{IV}$ BUS

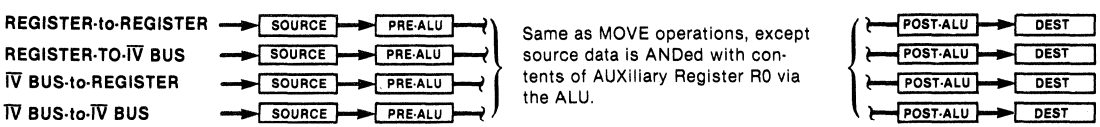
SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "S_1" and "S_0" fields of instruction.	Right rotate and mask as specified by "S_0" and "L" fields of instruction.	No operation.	Shift and merge as specified by "D_0" and "L" fields of instruction.	Variable length field of $\overline{IV}$ bus—Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) as specified by "D_0" and "D_1" fields of instruction.

ADD OPERATIONS



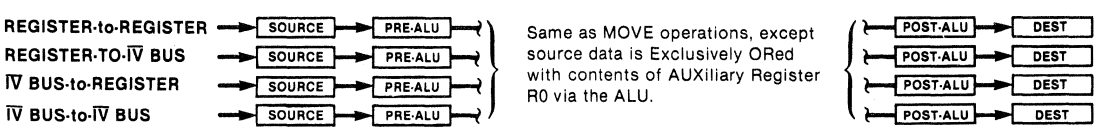
Same as MOVE operations, except source data is ADDed to contents of AUXiliary Register R0 via the ALU; if appropriate, Overflow Register R10 (OVF) is also set.

AND OPERATIONS



Same as MOVE operations, except source data is ANDed with contents of AUXiliary Register R0 via the ALU.

EXCLUSIVE OR (XOR) OPERATIONS



Same as MOVE operations, except source data is Exclusively ORed with contents of AUXiliary Register R0 via the ALU.

EXECUTE (XEC) OPERATIONS

XEC, REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
R0-R17 as specified by "S" field of instruction.	No operation.	Add source data to 8-bit field specified by instruction literal ($0 \leq J \leq 377_8$).	No operation.	Replace 8 LSB of Address Register* with 8-bit sum from ALU. *PGM CTR unchanged.

XEC, $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
Left or Right Bank of $\overline{IV}$ bus as specified by "S" field of instruction.	Rotate and mask as specified by "S_0" and "L" fields of instruction.	Add masked field of source data to 5-bit literal specified by "J" field of instruction ($0 \leq J \leq 37_8$).	No operation.	Replace 5 LSB of Address Register* with 5-bit sum from ALU. *PGM CTR unchanged.

NON-ZERO TRANSFER (NZT) OPERATIONS

NZT, REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DESTINATION
R0-R17 as specified by "S" field of instruction.	No operation.	Test contents of source register for all zeroes.	If S = 0, increment PC by 1; if S ≠ 0, replace 8 LSB of AR and PC with literal specified by "J" field of instruction.

NZT, $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DESTINATION
Left or Right Bank of $\overline{IV}$ bus as specified by "S" field of instruction.	Rotate and mask as specified by "S_0" and "L" fields of instruction.	Test contents of masked field for all zeroes.	If S = 0, increment PC by 1; if S ≠ 0, replace 5 LSB of AR and PC with literal specified by "J" field of instruction.

TRANSMIT (XMIT) OPERATIONS

XMIT, REGISTER

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
$0 \leq J \leq 377_8$ – value specified by "J" field of instruction.	No operation.	No operation.	No operation.	R0-R7, R11, R14-R17. Load 8-bit integer specified by "J" field into register specified by "D" field.

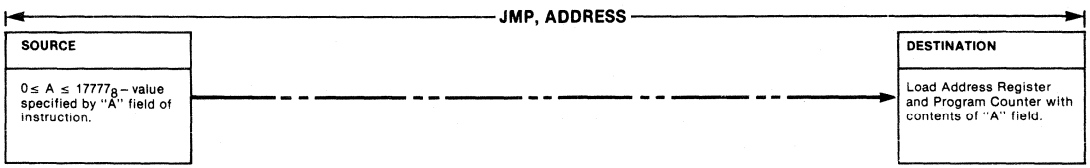
XMIT 8-BITS IMMEDIATE, $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
$0 \leq J \leq 377_8$ – value specified by "J" field of instruction.	No operation.	No operation.	No operation.	Left (R12) or Right (R13) Bank of $\overline{IV}$ bus as specified by "D" field of instruction.

XMIT VARIABLE-BIT FIELD IMMEDIATE $\overline{IV}$ BUS

SOURCE	DATA PROCESSING (PRE-ALU)	ALU	DATA PROCESSING (POST-ALU)	DESTINATION
$0 \leq J \leq 37_8$ – value specified by "J" field of instruction.	No operation.	No operation.	Shift and merge source data as specified by "D_0" and "L" fields of instruction.	Left or Right Bank of $\overline{IV}$ bus as specified by "D" field of instruction.

JUMP (JMP) OPERATION



Program Storage Interface

As shown in Figure 3, program storage is connected to output address lines A_0 through A_{12} (A_{12} = LSB) and input instruction lines I_0 through I_{15}. An address output on A_0/A_{12} identifies one 16-bit instruction word in program storage. The instruction word is subsequently input on I_0/I_{15} and defines the MicroController operation which is to follow — one instruction word equals one completed operation. Any TTL-compatible memory can be used for program storage provided the worst-case access time is compatible with the instruction cycle time used for the application — see timing section for appropriate calculations.

I/O Interface and Control

An 8-bit bidirectional I/O bus, referred to as the Interface Vector ($\overline{IV}$) bus, provides a communication link between the MicroController and the two banks of I/O devices. The $\overline{LB}$ (Left Bank) and $\overline{RB}$ (Right Bank) control signals identify which bank is enabled; when both $\overline{LB}$ and $\overline{RB}$ are high (inactive), neither bank is enabled and the $\overline{IV}$ bus is inactive (three-state). A functional analysis of the Left and Right Bank signals is shown below:

$\overline{LB}$	$\overline{RB}$	FUNCTION
Low	Low	This state is not generated by the 8X305.
Low	High	Enable left bank devices.
High	Low	Enable right bank devices.
High	High	Disable all devices; $\overline{IV}$ bus is three-state.

Both data and I/O address information are multiplexed on the $\overline{IV}$ bus. The SC (Select Command) and WC (Write Command) signals distinguish between data and I/O address information as follows:

LB/RB	SC	WC	FUNCTION
High	Low	Low	The $\overline{IV}$ bus is three-state and not looking for input data.
Low	Low	Low	The $\overline{IV}$ bus is reading input data.
Low	Low	High	Data is being output.
Low	High	Low	Address is being output.
X	High	High	This condition is never generated.

Data Processing

Basically, the data processing path of the 8X305 consists of the Rotate/Mask logic, the Arithmetic Logic Unit (ALU), the Shift/ Merge functions, on-chip memory (sixteen 8-bit registers), and the bidirectional $\overline{IV}$ bus interface with its associated driver circuits and internal latches. The on-board memory and the $\overline{IV}$ bus are connected to both inputs and outputs of the ALU via internal 8-bit data paths — see Figure 1. Inputs to the ALU are preceded by right-rotate and data-mask functions; the ALU output is followed by the left-shift and merge operations. Depending on the desired operation, any one or all of the functions (Rotate/Mask/Shift/Merge) can operate on 8 bits of data in a single instruction cycle. For a summary of all data-processing capabilities, refer to BASIC OPERATIONS OF THE 8X305 described earlier in this data sheet.

Instruction Cycle

Each operation of the 8X305 is executed in a single instruction cycle. The instruction cycle is internally divided into four equal parts — each part being as short as 50 nanoseconds. Figure 4 shows the general functions that

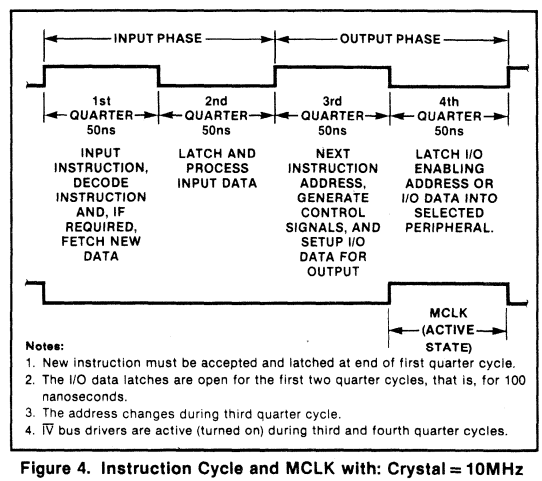


Figure 4. Instruction Cycle and MCLK with: Crystal = 10MHz and Cycle Time = 200 nanoseconds.

occur during each quarter cycle; specifics regarding minimum/maximum timing and other critical values are described later in this data sheet. During the first quarter cycle, a new instruction from program storage is input via I_0-I_{15} and decoded. If an I/O operation is indicated, new data is fetched from a specified internal register or via the $\overline{IV}$ bus. At the end of the first quarter cycle, the new instruction is latched into the instruction register.

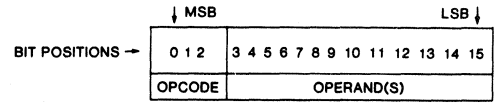
In the second quarter cycle, the I/O input data stabilizes and preliminary processing is completed; at the end of this quarter, the $\overline{IV}$ latches close and final processing can be accomplished, thus completing the input phase of the instruction cycle. During the third quarter cycle, the address for the next instruction is output to the instruction address bus, $\overline{IV}$ control signals are generated, and both data and destination are setup for the remainder of the output phase. During the fourth quarter cycle, a master clock signal (MCLK) generated by the 8X305 is used to latch either the I/O-enabling address or the I/O data into peripheral devices connected to the $\overline{IV}$ bus; MCLK can also be used to synchronize any external logic with timing circuits of the 8X305. To summarize the action, the first half of the instruction cycle deals primarily with input functions and the second half is mostly concerned with output functions.

INSTRUCTION SET

General Format and Operating Principles

The 16-bit instruction word (I_0 through I_{15}) from program storage is input to the instruction register (Figure 1) and is subsequently decoded to implement the events to occur during the current instruction cycle.

The general format for each instruction word is as follows:



The 3-bit operation code (OPCODE) define any one of eight classes of instructions; variations within each class are specified by the remaining thirteen operand bits. The eight instruction classes can be separated into two control areas — *data* and *program;* general functions within these areas are:

- Data Control —
  - ADD, AND, XOR — Arithmetic and Logic Operations
  - MOVE, XMIT — Movement of Data and Constants
- Program Control
  - XEC, NZT, JMP — Branch or Test

Instruction Fields

As shown in Table 1, each instruction word consists of an operation code (OPCODE) field and from one to three operand fields. The possible operand fields are: Source (S), Destination (D), Rotate/Length (R/L), Literal (J), and Address (A). The OPCODE and operand fields are described in the paragraphs that follow the table.

Table 1. FUNCTIONAL DESCRIPTION OF INSTRUCTION SET

INSTRUCTION WORD	DESCRIPTION	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE — SEE FIGURE 4		
		CONTROL SIGNAL	INPUT PHASE	OUTPUT PHASE
CLASS = **MOVE** OPCODE = **0** OPERATION = **(S) → D**				
Register-to-Register 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | R | D $S = 00_8$-17_8 $D = 00_8$-07_8, 11_8-17_8	Move content of internal register specified by S-field to internal register specified by D-field. Prior to the "MOVE" operation, right-rotate contents of internal source register by octal value (0 through 7) defined by the R-field.	SC	L	H if $D = 07_8$, 17_8
		WC	L	L
		$\overline{LB}$	H	L if $D = 07_8$
		$\overline{RB}$	H	L if $D = 17_8$
Register-to-$\overline{IV}$ Bus (Note) 0 1 2 | 3 4 5 6 7 | 8 9 10 | 11 12 13 14 15 OPCODE | S | L | D (D_1 : D_0) $S = 00_8$-17_8 $D = 20_8$-37_8	Move contents of internal register specified by the S-field to the $\overline{IV}$ bus. Before outputting on $\overline{IV}$ bus, data is shifted as specified by the least significant octal digit of the D-field and the bits specified by the L-field are merged with the latched I/O data.	SC	L	L
		WC	L	H
		$\overline{LB}$	L if $D = 20_8$-27_8	L if $D = 20_8$-27_8
		$\overline{RB}$	L if $D = 30_8$-37_8	L if $D = 30_8$-37_8

Table 1. FUNCTIONAL DESCRIPTION OF INSTRUCTION SET (Continued)

INSTRUCTION WORD	DESCRIPTION	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE — SEE FIGURE 3		
		CONTROL SIGNAL	INPUT PHASE	OUTPUT PHASE
CLASS = MOVE OPCODE = 0 OPERATION = (S) → D				
$\overline{IV}$ Bus-to-Register (Note)	Move right-rotated $\overline{IV}$ bus (source) data specified by the S-field to internal register specified by the D-field. The L-field specifies the length of source data starting from the LSB-position and, if less than 8 bits, the remaining bits are filled with zeros.	SC	L	H if $D = 07_8$, 17_8
		WC	L	L
		$\overline{LB}$	L if $S = 20_8$-27_8	L if $D = 07_8$
		$\overline{RB}$	L if $S = 30_8$-37_8	L if $D = 17_8$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | L | D S_1 ⋮ S_0				
$S = 20_8$-37_8 $D = 00_8$-07_8, 11_8-17_8				
$\overline{IV}$ Bus-to-$\overline{IV}$ Bus (Note)	Move right-rotated $\overline{IV}$ bus (source) data specified by the S-field to the I/O latches. Before outputting on $\overline{IV}$ bus, shift data as specified by the D-field; then merge source and latched I/O data as specified by the L (length) field.	SC	L	L
		WC	L	H
		$\overline{LB}$	L if $S = 20_8$-27_8	L if $D = 20_8$-27_8
		$\overline{RB}$	L if $S = 30_8$-37_8	L if $D = 30_8$-37_8
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | L | D S_1 ⋮ S_0 | D_1 ⋮ D_0				
$S = 20_8$-37_8 $D = 20_8$-37_8				
CLASS = ADD OPCODE = 1 OPERATION = (S) + (AUX) → D				
Same as MOVE instruction class	Same as MOVE instruction class except that contents of AUX (R0) register are ADDed to the source data. If there ia a "carry" from MSB, then R10 (OVF) = 1 (overflow), otherwise OVF = 0.	Same as MOVE instruction class		
CLASS = AND OPCODE = 2 OPERATION = (S) ∧ (AUX) → D				
Same as MOVE instruction class	Same as MOVE instruction class except that contents of AUX (R0) register are ANDed with source data.	Same as MOVE instruction class		
CLASS = XOR OPCODE = 3 OPERATION = (S) ⊕ (AUX) → D				
Same as MOVE instruction class	Same as MOVE instruction class except that contents of AUX (R0) register are Exclusively ORed with source data.	Same as MOVE instruction class		
CLASS = XEC OPCODE = 4 OPERATION = Refer to Description				
Register Immediate	Execute instruction at current page address offset by J (literal) + (S). Return to normal instruction flow unless a branch is encountered.	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | J	Execute instruction at an address determined by replacing the low-order 8 bits of the Address Register with the following derived sum: Value of literal (J-field) plus contents of internal register specified by S-field The PC is not incremented and the overflow status (OVF) is not changed.			
$S = 00_8$-17_8 $J = 000_8$-377_8				
$\overline{IV}$ Bus Immediate (Note)	Execute instruction at an address determined by replacing the low-order 5 bits of Address Register with the following derived sum: 5-bit value of literal (J-field) plus value of rotated source data specified by S-field. The L-field specifies the length of source data starting from the LSB position and, if less than 8 bits, the remaining bits are filled with zeros; the Program Counter is not incremented and the overflow status (OVF) is not changed.	SC	L	L
		WC	L	L
		$\overline{LB}$	L if $S = 20_8$-27_8	H
		$\overline{RB}$	L if $S = 30_8$-37_8	H
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | L | J S_1 ⋮ S_0				
$S = 20_8$-37_8 $J = 00_8$-37_8				
CLASS = NZT OPCODE = 5 OPERATION = Refer to Description				
Register Immediate	If data specified by the S-field is not equal to zero, jump to current page address offset by value of J-field; otherwise, increment the Program Counter.	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S | J	If contents of internal register specified by S-field is non-zero, transfer to address determined by replacing the low-order 8 bits of Address Register and Program Counter with "J", otherwise, increment PC.			
$S = 00_8$-17_8 $J = 000_8$-377_8				

Table 1. FUNCTIONAL DESCRIPTION OF INSTRUCTION SET (Concluded)

INSTRUCTION WORD	DESCRIPTION	STATE OF CONTROL SIGNAL DURING INSTRUCTION CYCLE — SEE FIGURE 3		
		CONTROL SIGNAL	INPUT PHASE	OUTPUT PHASE
CLASS = **NZT** OPCODE = **5** OPERATION = **Refer to Description**				
$\overline{IV}$ Bus Immediate (Note) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | S (S_1 : S_0) | L | J $S = 20_8$-37_8 $J = 00_8$-37_8	If right-rotated and masked IV bus is non-zero, transfer to address determined by replacing low-order 5 bits of Address Register and Program Counter with "J", otherwise, increment PC. (The L-field specifies the length of source I/O data starting from the LSB-position and, if less than 8 bits, the remaining bits are filled with zeros.)	SC	L	L
		WC	L	L
		$\overline{LB}$	L if $S = 20_8$-27_8	H
		$\overline{RB}$	L if $S = 30_8$-37_8	H
CLASS = **XMIT** OPCODE = **6** OPERATION = **J → D**				
XMIT, Register 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | J $D = 00_8$-06_8, 11_8, 14_8-16_8 $J = 000_8$-377_8	Store 8-bit value specified by "J" into register specified by "D".	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H
XMIT, $\overline{IV}$ Bus Address 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | J $D = 07_8$, 17_8 $J = 000_8$-377_8	Enable I/O device on the bank specified by "D", whose address is the 8-bit integer specified by "J". Address "J" is stored in register "D"	SC	L	H
		WC	L	L
		$\overline{LB}$	H	L if $D = 07_8$
		$\overline{RB}$	H	L if $D = 17_8$
XMIT 8 Bits Immediate, $\overline{IV}$ Bus (Note) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D | J $D = 12_8$-13_8 $J = 000_8$-377_8	Store value of 8-bit integer in the previously enabled I/O port, at the bank destination ($\overline{LB}$ or $\overline{RB}$) specified by "D". Contents of R12 or R13 remain unchanged.	SC	L	L
		WC	L	H
		$\overline{LB}$	H	L if $D = 12_8$
		$\overline{RB}$	H	L if $D = 13_8$
XMIT Variable Bit Field Immediate, $\overline{IV}$ Bus (Note) 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | D (D_1 : D_0) | L | J $D = 20_8$-37_8 $J = 00_8$-37_8	Transmit Least Significant "L" bits of "J" field to "L-bit" field of $\overline{IV}$ bus specified by "D"; if "L" is greated than 5 bits, the MSB bits of destination field is filled with zeros.	SC	L	L
		WC	L	H
		$\overline{LB}$	L if $D = 20_8$-27_8	L if $D = 20_8$-27_8
		$\overline{RB}$	L if $D = 30_8$-37_8	L if $D = 30_8$-37_8
CLASS = **JMP** OPCODE = **7** OPERATION = **Refer to Description**				
Address Immediate 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 OPCODE | A $A = 00000_8$-17777_8	Jump to address in program storage specified by A-field; this address is loaded into the Address Register and the Program Counter.	SC	L	L
		WC	L	L
		$\overline{LB}$	H	H
		$\overline{RB}$	H	H

Note:
S_0 specifies the LSB of rotated input data field
S_1 specifies the bank of $\overline{IV}$ bus from which source data will be input
D_0 specifies bit position in I/O device with which LSB of processed data will be aligned and
D_1 specifies the bank of $\overline{IV}$ bus which will be the destination.

Operations Code Field. The 3-bit OPCODE field specifies one of eight classes of 8X305 instructions; octal designations for this field and operands for each instruction class are shown in the preceding table.

Source (S) and Destination (D) Fields. The 5-bit "S" and "D" fields specify the source and destination, respectively, for whatever operation is defined by the OPeration CODE. The "S" and/or "D" fields can specify an internal 8X305 register or any one-to-eight bit field within an I/O device; octal values and source/destination field assignments for all internal registers are shown in Table 2.

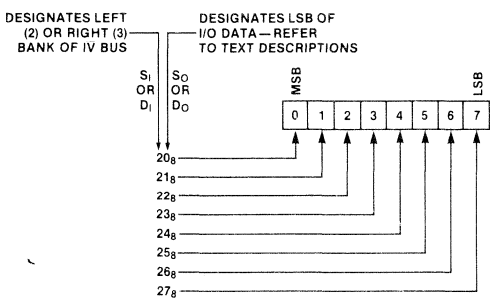
Table 2. OCTAL ADDRESSES AND SOURCE/DESTINATION FIELDS FOR 8X305 REGISTERS

ADDRESS	REGISTER DESIGNATION	SOURCE	DESTI-NATION	ADDRESS	REGISTER DESIGNATION	SOURCE	DESTI-NATION
00_8	R0 (AUX)—General purpose register	X	X	10_8	R10 (OVF—Overflow register)	X	
01_8	R1—General purpose register	X	X	11_8	R11—General purpose register	X	X
02_8	R2—General purpose register	X	X	12_8	R12—General purpose register (Note)	X	X
03_8	R3—General purpose register	X	X	13_8	R13—General purpose register (Note)	X	X
04_8	R4—General purpose register	X	X	14_8	R14—General purpose register	X	X
05_8	R5—General purpose register	X	X	15_8	R15—General purpose register	X	X
06_8	R6—General purpose register	X	X	16_8	R16—General purpose register	X	X
07_8	R7—Special purpose register (refer to next paragraph)	X	X	17_8	R17—Special purpose register (refer to next paragraph)	X	X

Note:
R12 and R13 function as general purpose working registers for all operations except transmit (XMIT). During a transmit instruction where R12 or R13 is the destination, the 8-bit "J" field is immediately transferred to the $\overline{IV}$ bus; for this operation, the contents of the designated register remain unchanged.

In instructions where $R7_8$ (IVL) or $R17_8$ (IVR) is specified as the destination, the 8-bit value is output on the $\overline{IV}$ bus as an I/O device address or memory location; register R7 selects the Left Bank and register R17 selects the Right Bank. The results are also stored into the specified internal register ($R7_8$ or $R17_8$) and may later be accessed as source data. When the $\overline{IV}$ bus is specified as a source and/or destination, the "S" and "D" fields are split into two parts, that is,

- Source (S) = S_1, S_0 and Destination (D) = D_1, D_0 where,
  S_0 specifies the LSB of rotated input data field
  S_1 specifies the bank of $\overline{IV}$ bus from which source data will be input
  D_0 specifies bit position in I/O device with which LSB of processed data will be aligned and
  D_1 specifies the bank of $\overline{IV}$ bus which will be the destination

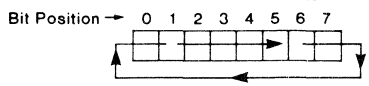


Notes:
1. The field length of 0-to-8 bits is specified by the "L" field.
2. For the Right Bank, 30_8-37_8 perform equivalent I/O functions.

Rotate (R) and Length (L) Field. The 3-bit R/L field performs one of two functions, specifying either the field length (L) for I/O operations or a right-rotate (R) for internal operations. For a given instruction, the specified function depends upon the contents of the Source (S) and Destination (D) fields.

When an internal register is specified by both the source and destination fields, the "R" field is invoked and it specifies a right-rotate of the data specified in the "S" field — see accompanying diagram. The source-register data (up to 8 bits) is right-rotated during the "input phase" of the instruction cycle (Figure 4) and this function is always performed prior to any ALU operation. (*Note: The right-rotate function is implemented on the bus and not in the source register.*)

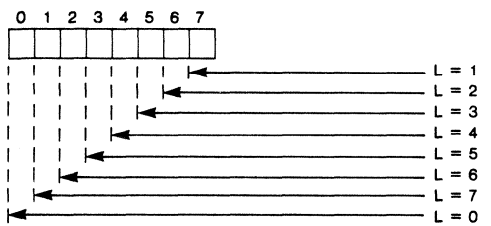
RIGHT-ROTATE FUNCTION



When either or both of the source and destination fields specify a variable-length I/O data field, the "L" field specifies the length of the I/O data field — see following diagram. If the source field specifies an $\overline{IV}$ address (20_8-37_8) and the destination field specifies an internal register (00_8-07_8, 11_8-17_8), the "L" field specifies the length of source data; the source data is formed by right-rotating the $\overline{IV}$ bus data according to the source address and then masking result as specified by the "L" field. If length is less than 8 bits, all remaining bits are set to zero prior to processing data in the ALU. If the source field

specifies an internal register (00_8-17_8) and the destination field specifies $\overline{IV}$ bus data (20_8-37_8), the "L" field specifies the length of the destination data. To form the destination data, the ALU output is left-shifted according to the destination address and then masked to the required length — see $\overline{IV}$ DATA LENGTH SPECIFICATION. The destination data is merged with data in the I/O latches to finalize the $\overline{IV}$ bus data. Hence, a one-to-eight bit destination data field can be inserted into the existing 8-bit I/O port without modifying surrounding bits. If both the source and destination fields specify $\overline{IV}$ bus data (20_8-37_8), the "L" field specifies the length of both the source and destination data.

$\overline{IV}$ DATA LENGTH SPECIFICATION
(No Rotate Function Specified)



To form the source data, the $\overline{IV}$ bus input data is right-rotated according to the source address and then masked to the required length—see $\overline{IV}$ DATA LENGTH SPECIFICATION. If length is less than 8 bits, all remaining bits are set to zero before processing in the ALU. To form the destination data, the ALU output is left-shifted according to the destination address and masked to the required length specification. The destination data is then merged into the $\overline{IV}$ bus data that was used to obtain the source; thus, if the source and destination addresses are on the same bank, the $\overline{IV}$ bus data written to the destination I/O Port appears unmodified, except for bits changed during the shift-and-mask operations. If the source and destination addresses refer to different banks, the destination I/O Port is changed to contain the contents of the source I/O Port in those bit positions not affected by the destination data.

J Field. The 5-bit or 8-bit "J" field is used to load a literal value (contained in the instruction) into a register, into a variable I/O data field, or to modify the low-order bits of the Program Counter. The bit length of the "J" field is implied by the "S" and "L" fields in the XEC, NZT, and XMIT instructions, based on the following conditions:

- When the Source (S) field specifies an internal register, the literal value of the "J" field is an 8-bit binary number.
- When the Source (S) field specifies a variable I/O data field, the literal value of the "J" field is a 5-bit binary number.

A Field. The 13-bit "A" field is an address field which allows the 8X305 to directly branch to any of the 8192 locations in Program Storage memory.

Formation of Instruction Address

The Address Register and Program Counter are used to generate addresses for accessing an instruction from program storage. The instruction address is formed in one of the following ways:

- For all except the JMP, XEC, and a "satisfied" NZT instruction, the Program Counter is incremented by one and placed in the Address Register.
- For the JMP instruction, the 13-bit "A" field contained in the JMP instruction word replaces the contents of both the Address Register and the Program Counter.
- For the XEC instruction, the Address Register is loaded with bits from the Program Counter modified as follows:

  XEC using $\overline{IV}$ Bus Data — low-order 5 bits of ALU output replaces counterpart bits in Address Register

  XEC using Data from Internal Register — low-order 8 bits of ALU output replaces counterpart bits in Address Register

  The Program Counter is not modified for either of the above conditions.
- For a "satisfied" NZT instruction, the low-order 5 bits (NZT source is $\overline{IV}$ bus data) or low-order 8 bits (NZT source is an internal register) of both the Address Register and Program Counter are loaded with the literal value specified by the "J" field of instruction word.

Data Addressing

The source and/or destination addresses of the data to be operated upon are specified as part of the instruction word. As shown earlier, source/destination addresses are specified using a 5-bit code (00_8-37_8). When the most significant octal digit is a "0" or "1", the source and/or destination address is an internal register; if the most significant digit is a 2 or 3, an $\overline{IV}$ bus operation is indicated — 2 specifying a Left-Bank ($\overline{LB}$) operation and 3 specifying a Right-Bank ($\overline{RB}$) operation. The least significant octal digit (0 through 7) indicates either a specific internal register address or positioning information for the least significant bit when specifying $\overline{IV}$ bus data. Referring to Table 1, AUXiliary register R0 (00_8) is the implied source

of the second argument for the ADD, AND, and XOR operations. IVL register R7 and IVR register R17 (destination addresses 07_8 and 17_8, respectively) provide a means of routing enabling address information to I/O peripherals. With IVL or IVR specified as the destination address, data is placed on the $\overline{IV}$ bus during the output phase of the instruction cycle; simultaneously, a Select Command (SC) is generated to inform all I/O devices that information on the $\overline{IV}$ bus is to be considered as an I/O address. Since the contents of IVL and IVR are preserved, either register may later be accessed as a source of data.

Control outputs $\overline{LB}$ and $\overline{RB}$ are used to partition I/O bus devices into two fields of 256 addresses. With $\overline{LB}$ in the active-low state and a source address of 20_8–27_8, the left bank of I/O devices are enabled during the input phase of the instruction cycle. With $\overline{RB}$ in the active-low state and a source address of 30_8–37_8, the right bank of devices are enabled. During the output phase, $\overline{LB}$ is low if the destination address is 07_8 or 20_8–27_8, whereas $\overline{RB}$ is low if the destination address is 17_8 or 30_8–37_8. Each address field ($\overline{LB}$ and $\overline{RB}$) can have a different I/O device selected, that is, data can be transferred from a device in one bank to a device in the other in one instruction cycle.

DESIGN PARAMETERS

Hardware design of an 8X305-based system largely consists of the following operations:

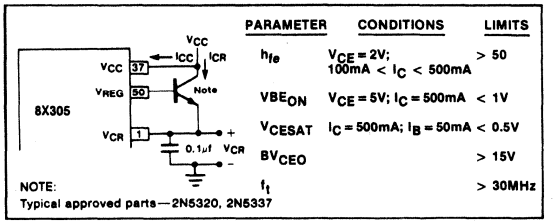
- Selecting and interfacing a Program Storage device — ROM, PROM, etc.
- Selecting and interfacing input/output devices — RAM, Ports, and other 8-bit addressable I/O devices.
- Choosing and implementing System Clock — Capacitor-Controlled, Crystal-Controlled, or Externally-Driven.
- Selection of an off-chip series-pass transistor.

All information required for easy implementation of these design requirements is provided under the following captions:

- Ordering Information
- Voltage Regulator
- DC Characteristics
- AC Characteristics
- Timing Considerations
- Clock Considerations
- $\overline{HALT/RESET}$ Logic

VOLTAGE REGULATOR

All internal logic of the 8X305 is powered by an on-chip voltage regulator that requires an external series-pass transistor. Electrical specifications for the off-chip power transistor and a typical hook-up are shown in the accompanying diagram. To minimize lead inductance, the transistor should be as close as possible to the 8X305 package and the emitter should be ac-grounded via a 0.1 microfarad ceramic capacitor.



DC CHARACTERISTICS (Commercial Part) $4.75V \leq V_{CC} \leq 5.25V$, $0°C \leq T_A \leq 70°C$

ABSOLUTE MAXIMUM RATINGS

Storage Temperature (T_{STG}) ratings are from -65 to $+150°C$

PIN	DESCRIPTION	RATING	UNIT	PIN	DESCRIPTION	RATING	UNIT
V_{CC}	Supply voltage	+7.0	V	All other pins	Logic input voltage	5.5	V
X1, X2	Crystal input voltage	2.0	V				

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT	COMMENTS
V_{CC}	Supply voltage		4.75	5.0	5.25	V	
V_{IH}	High level input voltage		0.6 2.0		2.0 5.5	V	X1 and X2 All other pins
V_{IL}	Low level input voltage				0.4 0.8	V	X1 and X2 All other pins
V_{OH}	High level output voltage	V_{CC} = min; $I_{OH} = -3mA$	2.4			V	
V_{OL}	Low level output voltage	V_{CC} = min; I_{OL} = 6mA V_{CC} = min; I_{OL} = 16mA			0.55 0.55	V	A_0 through A_{12} All other outputs
V_{CR}	Regulator voltage	V_{CC} = 5V		3.1 2.9		V	$T_A = 0°C$ $T_A = 70°C$
V_{IC}	Input clamp voltage	V_{CC} = min; $I_{IN} = -10mA$			−1.5	V	Crystal inputs X1 and X2 do not have internal clamp diodes
I_{IH}	High level input current	V_{CC} = max; V_{IH} = 0.6V V_{IH} = 4.5V			4.0 50	mA μA	X1 and X2 All other pins
I_{IL}	Low-level input current	V_{CC} = max; V_{IL} = 0.4V			−3 −0.2 −1.6 −0.4	mA	X1 and X2 $\overline{IV0}$-$\overline{IV7}$ I0-I15 $\overline{HALT}$ and $\overline{RESET}$
I_{OS}	Short circuit output current	V_{CC} = max; (Note: At any time, no more than one output should be connected to ground.)	−30		−140	mA	All output pins
I_{CC}	Supply current	V_{CC} = max			180 195	mA	$T_A = 70°C$ $T_A = 0°C$
I_{REG}	Regulator control	V_{CC} = 5.0V	−10		−25	mA	Max available base drive for series-pass transistor
I_{CR}	Regulator current	V_{CC} = max			200 230	mA	$T_A = 70°C$ $T_A = 0°C$

Notes:
1. Operating temperature ranges are guaranteed after thermal equilibrium has been reached.
2. All voltages measured with respect to ground terminal.

AC CHARACTERISTICS (Commercial Part) CONDITIONS: $4.75V \leq V_{CC} \leq 5.25V$; $0°C \leq T_A \leq 70°C$

LOADING: (See test circuits)

PARAMETER (Note 1)		LIMITS (INSTRUCTION CYCLE TIME = 200ns) Min	Typ	Max	LIMITS (INSTRUCTION CYCLE TIME > 200ns) Min	Typ	Max	UNITS	COMMENTS
T_{PC}	Processor cycle time	200			200			ns	
T_{CP}	X1 clock period	100			100			ns	
T_{CH}	X1 clock high time	50			50			ns	
T_{CL}	X1 clock low time	50			50			ns	
T_{MCL}	MCLK low delay	15		40	15		40	ns	
T_W	MCLK pulse width	40		60	$T_{4Q} - 10$		$T_{4Q} + 10$	ns	Note 2
T_{MODO}	Output driver turn on time MCLK falling edge	125		145	$T_{1Q} + T_{2Q} + 25$		$T_{1Q} + T_{2Q} + 45$	ns	Note 9

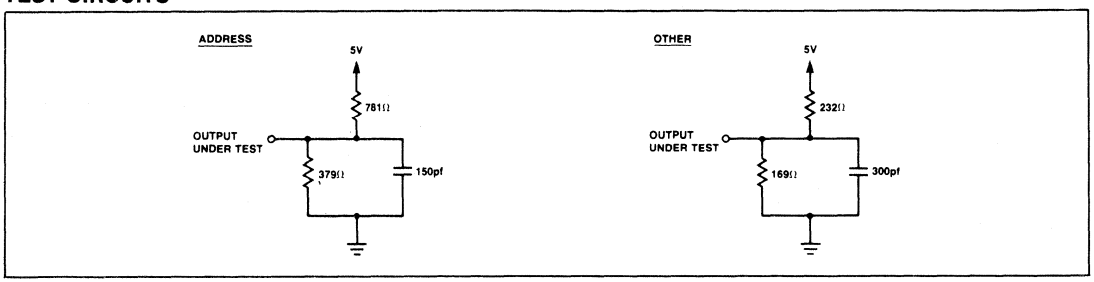
AC CHARACTERISTICS (Commercial Part) (Continued)

CONDITIONS: $4.75V \leq V_{CC} \leq 5.25V$; $0°C \leq T_A \leq 70°C$
LOADING: (See test circuits)

PARAMETER (Note 1)		LIMITS (INSTRUCTION CYCLE TIME = 200ns)			LIMITS (INSTRUCTION) CYCLE TIME > 200ns)			UNITS	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
T_{DI}	Output driver turn-on time (SC/WC rising edge)	20			20			ns	Note 10
T_{DD}	Input data to output data	85		105	85		105	ns	
T_{MHS}	MCLK falling edge to $\overline{HALT}$ falling edge			30			$T_{1Q} - 20$	ns	Note 2
T_{MHH}	$\overline{HALT}$ hold time (MCLK falling edge)	65			$T_{1Q} + 15$			ns	Note 2
T_{ACC}	Program storage access time			60				ns	
T_{IO}	I/O port output enable time ($\overline{LR}/\overline{RB}$ to valide $\overline{IV}$ data input)			30				ns	
T_{MAS}	MCLK falling edge to address stable			140			$T_{1Q} + T_{2Q} + 40$	ns	Notes 2, 3 & 4
T_{IA}	Instruction to address			140			$T_{2Q} + 90$	ns	Notes 2, 3 & 5
T_{IVA}	Input data to address			85			85	ns	Notes 3 & 6
T_{MIS}	MCLK falling edge to instruction stable			30			$T_{1Q} - 20$	ns	Notes 2 & 10
T_{MIH}	Instruction hold time (MCLK falling edge)	55			$T_{1Q} + 5$			ns	Notes 2 & 8
T_{MWH}	MCLK falling edge to SC/WC rising edge	105		125	$T_{1Q} + T_{2Q} + 5$		$T_{1Q} + T_{2Q} + 25$	ns	Note 2
T_{MWL}	MCLK falling edge to SC/WC falling edge	5		15	5		15	ns	
T_{MIBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Input phase)	10		25	10		25	ns	
T_{IIBS}	Instruction to $\overline{LB}/\overline{RB}$ (Input phase)			25			25	ns	
T_{MOBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Output phase)	115		145	$T_{1Q} + T_{2Q} + 15$		$T_{1Q} + T_{2Q} + 45$	ns	Note 2
T_{MIDS}	MCLK falling edge to input data stable			55			$T_{1Q} + T_{2Q} - 45$	ns	Note 2
T_{MIDH}	Input data hold time (MCLK falling edge)	115			$T_{1Q} + T_{2Q} + 15$			ns	Note 2
T_{MODH}	Output data hold time (MCLK falling edge)	11			11			ns	
T_{MODS}	Output data stable (MCLK falling edge)	130		150	$T_{1Q} + T_{2Q} + 30$		$T_{1Q} + T_{2Q} + 50$	ns	Note 2

NOTES:
1. X1 and X2 inputs are driven by an external pulse generator with an amplitude of 1.5 volts; all timing parameters are measured at this voltage level.
2. Respectively, T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q} represent time intervals for the first, second, third, and fourth quarter cycles.
3. Capacitive loading for the address bus is 150 picofarads.
4. T_{MAS} is obtained by forcing a valid instruction and an I/O bus input to occur earlier than the specified minimum set up time.
5. T_{IA} is obtained by forcing a valid instruction input to occur earlier than the minimum set up time.
6. T_{IVA} is obtained by forcing a valid I/O bus input to meet the minium set up time.
7. T_{MIS} represents the setup time required by internal latches of the 8X305. In system applications, the instruction input may have to be valid before the worst-case set up time in order for the system to respond with a valid I/O bus input that meets the I/O bus input set up time (T_{IDS} and T_{MIDS}).
8. T_{MIH} represents the hold time required by internal latches of the 8X305. To generate proper $\overline{LB}/\overline{RB}$ signals, the instruction must be held valid until the address bus changes.
9. The minimum figure for these parameters represents the earliest time that I/O bus output drivers of the 8X305 will turn on.
10. This parameter represents the latest time that the output drivers of the input device should be turned off.

TEST CIRCUITS



ABSOLUTE MAXIMUM RATINGS

Storage Temperature (T_{STG}) ratings are from −65 to +150°C

PIN	DESCRIPTION	RATING	UNIT	PIN	DESCRIPTION	RATING	UNIT
V_{CC}	Supply voltage	+7.0	V	All other pins	Logic input pins	5.5	V
X1, X2	Crystal input voltage	2.0	V				

DC CHARACTERISTICS (Military Part)

$4.5V \le V_{CC} \le 5.5V$, $-55°C \le T_C \le +125°C$

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT	COMMENTS
V_{CC}	Supply voltage		4.5	5.0	5.5	V	
V_{IH}	High level input voltage		0.6 2.0		2.0	V	X1 and X2 All other pins
V_{IL}	Low level input voltage				0.4 0.8	V	X1 and X2 All other pins
V_{OH}	High level output voltage	V_{CC} = min; I_{OH} = −3mA	2.4			V	
V_{OL}	Low level output voltage	V_{CC} = min; I_{OL} = 6mA V_{CC} = min; I_{OL} = 16mA			0.55 0.55	V	A_0 through A_{12} All other outputs
V_{CR}	Regulator voltage	V_{CC} = 5V		3.5 3.1 2.6		V	T_C = −55°C T_C = 0°C T_C = 125°C
V_{IC}	Input clamp voltage	V_{CC} = min; I_{IN} = −10mA			−1.5	V	Crystal inputs X1 and X2 do not have internal clamp diodes.
I_{IH}	High level input current	V_{CC} = max; V_{IH} = 0.6V V_{IH} = 4.5V			4.0 50	mA μA	X1 and X2 All other pins
I_{IL}	Low-level input current	V_{CC} = max; V_{IL} = 0.4V			−3 −0.3 −1.6 −0.4	mA	X1 and X2 $\overline{IV0}$-$\overline{IV7}$ I0-I15 $\overline{HALT}$ and $\overline{RESET}$
I_{OS}	Short circuit output current	V_{CC} = max; (Note: At any time, no more than one output should be connected to ground.)	−30		−140	mA	All output pins
I_{CC}	Supply current	V_{CC} = max			175 205	mA	T_C = 125°C T_C = −55°C
I_{REG}	Regulator control	V_{CC} = 5.0V	−10		−25	mA	Max available base drive for series-pass transistor
I_{CR}	Regulator current	V_{CC} = max			180 260	mA	T_C = 125°C T_C = −55°C

NOTES:
1. Operating temperature ranges are guaranteed after thermal equilibrium has been reached.
2. All voltages measured with respect to ground terminal.

AC CHARACTERISTICS (Military Part)

CONDITIONS: $4.5V \leq V_{CC} \leq 5.5V$; $-55°C \leq T_C \leq 125°C$
LOADING: (See test circuits)

PARAMETER (Note 1)		LIMITS (INSTRUCTION CYCLE TIME = 250ns)			LIMITS (INSTRUCTION) CYCLE TIME > 250ns)			UNITS	COMMENTS
		Min	Typ	Max	Min	Typ	Max		
T_{PC}	Processor cycle time	250			250			ns	
T_{CP}	X1 clock period	125			125			ns	
T_{CH}	X1 clock high time	62			62			ns	
T_{CL}	X1 clock low time	62			62			ns	
T_{MCL}	MCLK low delay	15		40	15		40	ns	
T_W	MCLK pulse width	47		72	$T_{4Q}-15$		$T_{4Q}+10$	ns	Note 2
T_{MODO}	Output driver turn-on time (MCLK falling edge)	145		175	$T_{1Q}+T_{2Q}+20$		$T_{1Q}+T_{2Q}+50$	ns	Note 9
T_{DI}	Output driver turn-on time (SC/WC rising edge)	20			20			ns	Note 10
T_{DD}	Input data to output data	80		115	80		115	ns	
T_{MHS}	MCLK falling edge to $\overline{HALT}$ falling edge			40			$T_{1Q}-22$	ns	Note 2
T_{MHH}	$\overline{HALT}$ hold time (MCLK falling edge)	80			$T_{1Q}+18$			ns	Note 2
T_{ACC}	Program storage access time			90				ns	
T_{IO}	I/O port output enable time ($\overline{LB}/\overline{RB}$ to valid $\overline{IV}$ data input)			40				ns	
T_{MAS}	MCLK falling edge to address stable			160			$T_{1Q}+T_{2Q}+35$	ns	Notes 2, 3 & 4
T_{IA}	Instruction to address			160			$T_{2Q}+98$	ns	Notes 2, 3 & 5
T_{IVA}	Input data to address			90			90	ns	Notes 3 & 6
T_{MIS}	MCLK falling edge to instruction stable			40			$T_{1Q}-22$	ns	Notes 2 & 10
T_{MIH}	Instruction hold time (MCLK falling edge)	70			$T_{1Q}+8$			ns	Notes 2 & 8
T_{MWH}	MCLK falling edge to SC/WC rising edge	127		154	$T_{1Q}+T_{2Q}+2$		$T_{1Q}+T_{2Q}+29$	ns	Note 2
T_{MWL}	MCLK falling edge to SC/WC falling edge	5		25	5		25	ns	
T_{MIBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Input phase)	10		35	10		35	ns	
T_{IIBS}	Instruction to $\overline{LB}/\overline{RB}$ (Input phase)			30			30	ns	
T_{MOBS}	MCLK falling edge to $\overline{LB}/\overline{RB}$ (Output phase)	140		170	$T_{1Q}+T_{2Q}+15$		$T_{1Q}+T_{2Q}+45$	ns	Note 2
T_{MIDS}	MCLK faling edge to input data stable			75			$T_{1Q}+T_{2Q}-50$	ns	Note 2
T_{MIDH}	Input data hold time (MCLK falling edge)	140			$T_{1Q}+T_{2Q}+15$			ns	Note 2
T_{MODH}	Output data hold time (MCLK falling edge)	11			11			ns	
T_{MODS}	Output data stable (MCLK falling edge)	150		180	$T_{1Q}+T_{2Q}+25$		$T_{1Q}+T_{2Q}+55$	ns	Note 2

NOTES:
1. X1 and X2 inputs are driven by an external pulse generator with an amplitude of 1.5 volts; all timing parameters are measured at this voltage level.
2. Respectively, T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q} represent time intervals for the first, second, third, and fourth quarter cycles.
3. Capacitive loading for the address bus is 150 picofarads.
4. T_{MAS} is obtained by forcing a valid instruction and an I/O bus input to occur earlier than the specified minimum set up time.
5. T_{IA} is obtained by forcing a valid instruction input to occur earlier than the minimum set up time.
6. T_{IVA} is obtained by forcing a valid I/O bus input to meet the minium set up time.
7. T_{MIS} represents the setup time required by internal latches of the 8X305. In system applications, the instruction input may have to be valid before the worst-case set up time in order for the system to respond with a valid I/O bus input that meets the I/O bus input set up time (T_{IDS} and T_{MIDS}).
8. T_{MIH} represents the hold time required by internal latches of the 8X305. To generate proper $\overline{LB}/\overline{RB}$ signals, the instruction must be held valid until the address bus changes.
9. The minimum figure for these parameters represents the earliest time that I/O bus output drivers of the 8X305 will turn on.
10. This parameter represents the latest time that the output drivers of the input device should be turned off.

TIMING CONSIDERATIONS (Commercial Part)

As shown in the AC CHARACTERISTICS table for the commercial part, the minimum instruction cycle time is 200 nanoseconds; whereas, the maximum is determined by the on-chip oscillator frequency and can be any value the user chooses. With an instruction cycle time of 200 nanoseconds, the part can be characterized in terms of absolute values; these are shown in the first "LIMITS" column of the table. When the instruction cycle time is greater than 200 nanoseconds, certain parameters are cycle-time dependent; thus, these parameters are specified in terms of the four quarter cycles (T_{1Q}, T_{2Q}, T_{3Q}, and T_{4Q}) that make up one instruction cycle — see 8X305 TIMING DIAGRAM. As the time interval for each instruction cycle increases (becomes greater than 200 nanoseconds), the delay for all parameters that are cycle-time dependent is likewise increased. In some cases, these delays have a significant impact on timing relationships and other areas of systems design; subsequent paragraphs describe these timing parameters and reliable methods of calculation.

Timing parameters for the 8X305 are normally measured with reference to MCLK.

System determinants for the instruction cycle time are:

- Propagation delays within the 8X305
- Access time of Program Storage
- Enable time of the I/O port

Normally, the instruction cycle time is constrained by one or more of the following conditions:

Condition 1 — Instruction or MCLK to $\overline{LB}/\overline{RB}$ (input phase) plus I/O port access time (TIO) ≤ $\overline{IV}$ data set up time (Figure 5a).

Condition 2 — Program storage access time (TACC) plus instruction to $\overline{LB}/\overline{RB}$ (input phase) plus I/O port access time (TIO) plus $\overline{IV}$ data (input phase) to address ≤ instruction cycle time (Figure 5b).

Condition 3 — Program storage access time plus instruction to address ≤ instruction cycle time (Figure 5c).

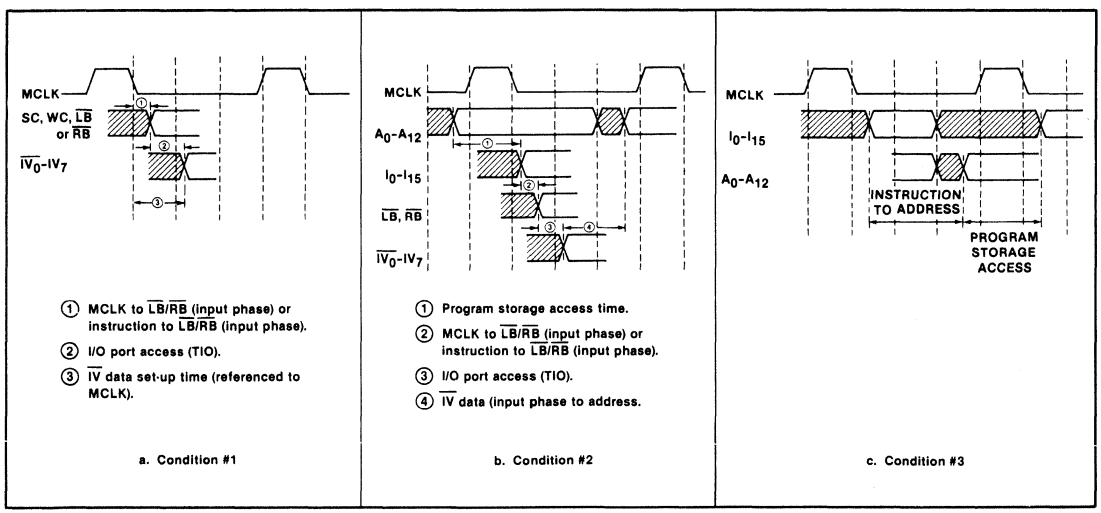
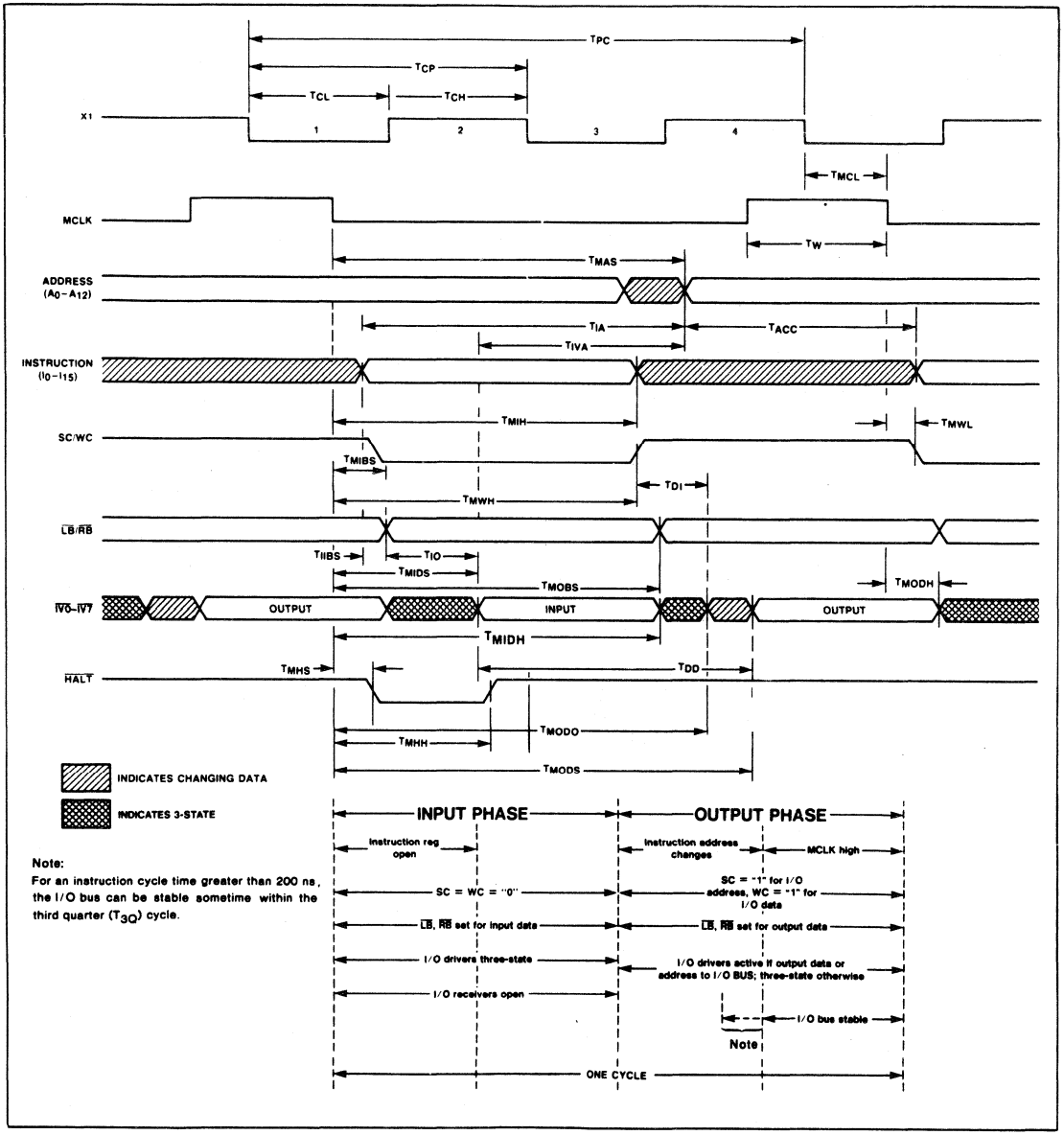


Figure 5. Constraints of 8X305 Instruction Cycle Time

8X305 TIMING DIAGRAM



Note:
For an instruction cycle time greater than 200 ns, the I/O bus can be stable sometime within the third quarter (T_{3Q}) cycle.

From condition #1 and with an instruction cycle time of 200ns, the I/O port access time (TIO) can be calculated as follows:

$$TMIBS + TIO \leq TMIDS$$
$$\text{transposing, } TIO \leq TMIDS = TMIBS$$
$$\text{substituting, } TIO \leq 55ns - 25ns$$
$$\text{result, } TIO \leq 30ns$$

Using 30ns for TIO, the constraint imposed by condition #1 can also be used to calculate the minimum cycle time:

$$TMIBS + TIO \leq TMIDS$$
$$\text{thus, } 25ns + 30ns \leq T_{1Q} + T_{2Q} - 45$$
$$25ns + 30ns \leq 1/2 \text{ cycle} - 45$$

therefore, the worst-case instruction cycle time is 200ns. With subject parameters referenced to X1, the same calculations are valid:

$$TIBS + TIO + TIDS \leq 1/2 \text{ cycle}$$
$$\text{thus, } 45ns + 30ns + 25ns \leq 1/2 \text{ cycle}$$

therefore, the worst-case instruction cycle time is again 200ns. From condition #2 and with an instruction cycle time of 200ns, the program storage access time can be calculated:

$$TACC + TIIBS + TIO + TIVA \leq 200ns$$
$$\text{transposing, } TACC \leq 200ns - TIIBS - TIO - TIVA$$
$$\text{substituting, } TACC \leq 200ns - 25ns - 30ns - 85ns$$
$$\text{thus, } TACC \leq 60ns$$

hence, for an instruction cycle time of 200ns, a program storage access time of 60ns is implied. The constraint imposed by condition #3 can be used to verify the maximum program storage access time:

$$TIA + TACC \leq \text{Instruction Cycle}$$
$$\text{thus, } TACC \leq 200ns - 140ns$$
$$\text{and, } TACC \leq 60ns$$

confirming that a program storage access time of 60ns is satisfactory.

For an instruction cycle time of 200ns and a program storage access time of 60ns (Condition #2/Figure 5b), the instruction should be valid at the falling edge of MCLK. This relationship can be derived by the following equation:

$$200ns - TMAS - TACC$$
$$= 200ns - 140ns - 60ns$$
$$= 0ns$$

It is important to note that, during the input phase, the beginning of a valid $\overline{LB}/\overline{RB}$ signal is determined by either the instruction to $\overline{LB}/\overline{RB}$ delay (TIIBS) or the delay from the falling edge of MCLK to $\overline{LB}/\overline{RB}$ (TMIBS). Assuming the instruction is valid at the falling edge of MCLK and adding the instruction-to-$\overline{LB}/\overline{RB}$ delay (TIIBS = 25ns), the $\overline{LB}/\overline{RB}$ signal will be valid 25ns after the falling edge of MCLK. With a fast program storage memory and with a valid instruction before the falling edge of MCLK — the $\overline{LB}/\overline{RB}$ signal will, due to the TMIBS delay, still be valid 25ns after the falling edge of MCLK. Using a worst-case instruction cycle time of 200ns, the user cannot gain a speed advantage by selecting a memory with faster access time. Under the same conditions, a speed advantage cannot be obtained by using an I/O port with fast access time (TIO) because the address bus will be stable 55ns (TAS) after the beginning of the third quarter cycle — no matter how early the $\overline{IV}$ data input is valid.

CLOCK CONSIDERATIONS

The on-chip oscillator and timing-generation circuits of the 8X305 can be controlled by any one of the following methods:

Capacitor — if timing is not critical
Crystal — if precise timing is required
External Drive — if application requires that the 8X305 be driven from a system clock

Capacitor Timing. A non-polarized ceramic or mica capacitor with a working voltage equal to or greater than 25 volts is recommended. The lead lengths of capacitor should be approximately the same and as short as possible; also, the timing circuits should not be in close proximity to external sources of noise. For various capacitor (C_X) values, the cycle time can be approximated as:

C_X (In pF)	APPROXIMATE CYCLE TIME
100	300ns
200	500ns
500	1.1μs
1000	2.0μs

Crystal Timing. When a crystal is used, the on-chip oscillator operates at the resonant frequency (f_o) of the crystal; the series-resonant quartz crystal connects to the 8X305 via pins 10 (X1) and 11 (X2). The lead lengths of the crystal should be approximately equal and as short as possible; also, the timing circuits should not be in close proximity to external sources of noise. The crystal should be hermetically sealed (HC type can) and have the following electrical characteristics:

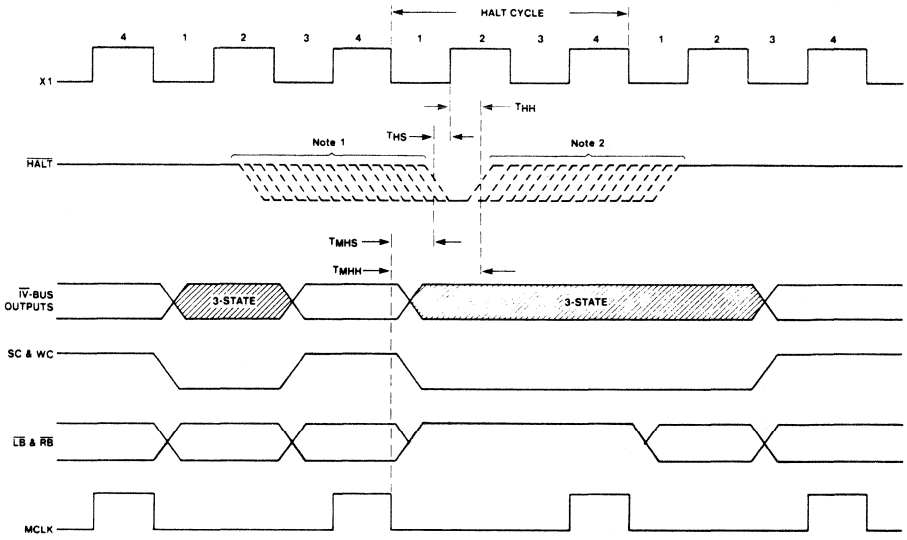
Type — Fundamental mode, series resonant
Impedance at Fundamental — 35 ohms maximum
Impedance at Harmonics and Spurs — 50 ohms minimum

The resonant frequency (f_o) of the crystal is related to the desired cycle time (T) by the equation: $f_o = 2/T$; thus, for a cycle time of 200 nanoseconds, $f_o = 10MHz$.

$\overline{HALT}$ Logic

The $\overline{HALT}$ signal is sampled via internal chip logic at the end of the first internal quarter of each instruction cycle. If, when sampled, the $\overline{HALT}$ signal is active-low, a halt is immediately executed and the current instruction cycle is terminated; however, the halt cycle does not inhibit MCLK nor does it affect any internal registers of the 8X305. As long as the $\overline{HALT}$ line is active-low, the SC and WC lines are low (inactive), the Left Bank ($\overline{LB}$)/Right Bank ($\overline{RB}$) signals are high (inactive), and the $\overline{IV}$ bus remains in the three-state mode of operation. Normal operation resumes at the next cycle in which $\overline{HALT}$ is high when sampled — see $\overline{HALT}$ TIMING DIAGRAM.

HALT TIMING DIAGRAM



Notes:

1. The $\overline{HALT}$ signal can switch from High to Low at any time during this interval.
2. The $\overline{HALT}$ signal can switch from Low to High at any time during this interval.

Timing Descriptions:

T_{HS}—set-up time from $\overline{HALT}$ to X1 (independent of instruction cycle time)

T_{HH}—hold time from X1 to $\overline{HALT}$ (independent of instruction cycle time)

T_{MHS}—set-up time from MCLK to $\overline{HALT}$ (dependent upon instruction cycle time)

T_{MHH}—hold time from MCLK to $\overline{HALT}$ (dependent upon instruction cycle time)

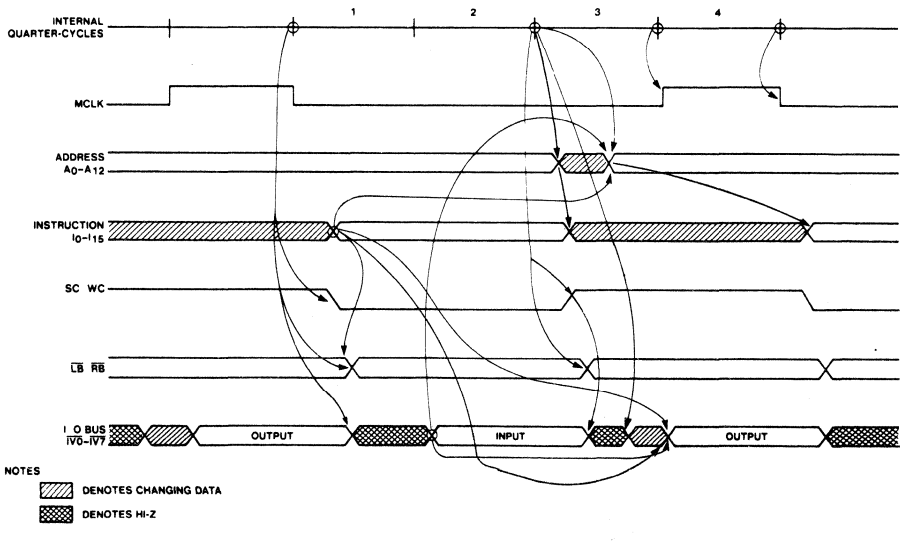


Figure 7. Timing Relationships of 8X305 I/O Signals

Using an External Clock. The 8X305 can be synchronized with an external clock by simply connecting appropriate drive circuits to the X1/X2 inputs. Figure 8 shows how the on-chip oscillator can be driven from the complementary outputs of a pulse generator. In applications where the MicroController must be driven from a master clock, the X1/X2 lines can be interfaced to TTL logic as shown in Figure 9.

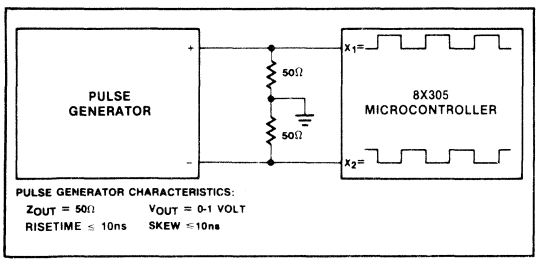


Figure 8. Clocking with a Pulse Generator

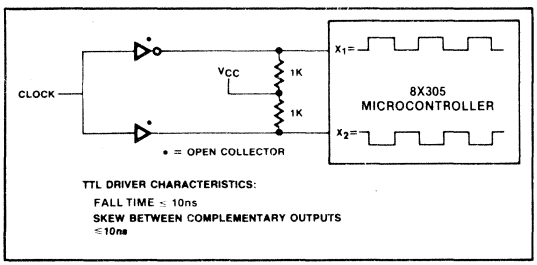


Figure 9. Clocking with TTL

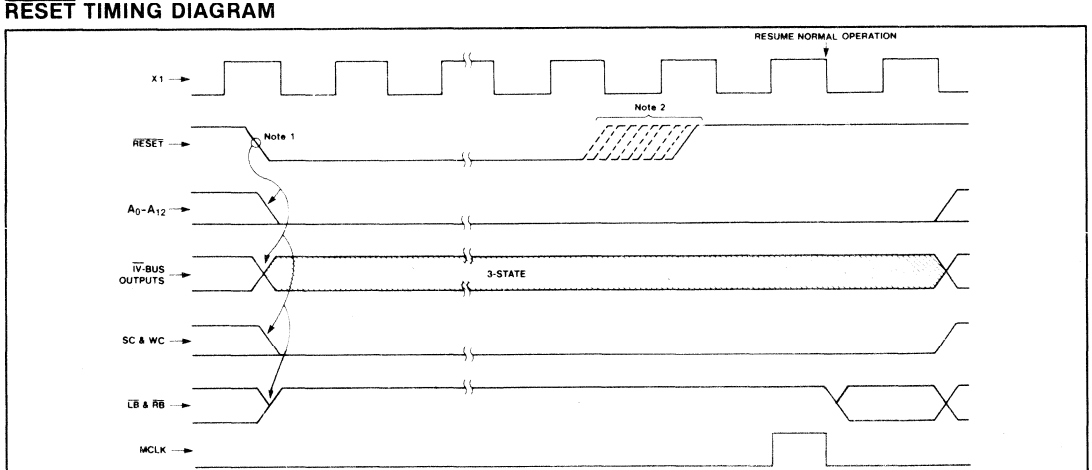
RESET Logic

RESET (pin 43) can be driven from a high (inactive) state to a low (active) state at any time with respect to the system clock, that is, the reset function is asynchronous. To ensure proper operation, RESET must be held low (active) for one full instruction time. When the line is driven from a high state to an active-low state, several events occur — the precise instant of occurrence is basically a function of the propagation delay for that particular event. As shown in the RESET TIMING DIAGRAM, these events are:

- The Program Counter and Address Register are set to address zero and remain in that state as long as the RESET line is low. Other than PC and AR, RESET does not affect other internal registers.
- The input/output (IV) bus goes three-state and remains in that condition as long as the RESET line is low.
- The Select Command and Write Command signals are driven low and remain low as long as the RESET line is low.
- The Left Bank/Right Bank (LB/RB) signals are forced high asynchronously for the period in which the RESET line is low.

During the time RESET is active-low, MCLK is inhibited; moreover, if the RESET line is driven low during the last two quarter cycles, MCLK may be shortened for that particular machine cycle. When RESET line is driven high (inactive)—one quarter to one full instruction cycle later, MCLK appears just before normal operation is resumed. The RESET/MCLK relationship is clearly shown by "B" in the timing diagram. As long as the RESET line is active-low, the HALT signal (described next) is not sampled by internal logic of the 8X305.

RESET TIMING DIAGRAM



INTERRUPT CONTROL COPROCESSOR

Originally published by Signetics January 1984

FEATURES

- **Three prioritized interrupts**
- **Subroutine handling capabilities**
- **4-level LIFO stack for return address storage**
- **Interrupt masking by software and hardware**
- **Stack full flag**
- **Directly compatible with 8X305 MicroController**
- **Bipolar ISL (Integrated Schottky Logic) and low-power Schottky technology**
- **Single +5 volt power supply**
- **0.6 inch, 40-pin DIP**

PRODUCT DESCRIPTION

The Signetics 8X310 Interrupt Control Coprocessor (ICC) supports the 8X305 MicroController in systems that are interrupt driven and those that require subroutine handling capabilities.

As shown in Figure 1, the ICC provides three prioritized interrupt request lines, INT 0 (highest priority), INT 1 and INT 2. A low-to-high transition applied to any of these input lines latches in an interrupt request which may be serviced when sampled by the ICC once each instruction cycle of the MicroController. When an interrupt request is serviced, the ICC forces the MicroController to jump to one of three fixed locations in program memory; instruction addresses 4, 5, and 6 correspond to INT 0, INT 1 and INT 2. At each of these addresses, the user programs a JMP instruction to another address where the user's interrupt service routine begins.

During interrupt servicing, the ICC also stores the proper return address into a four deep, Last-In-First-Out (LIFO) stack. At the conclusion of the interrupt service routine, the user program instructs the ICC to return to the main program at the location previously stored in the stack. The return operation is implemented by coding a special RETURN instruction which is decoded directly off the instruction bus by the ICC. There are five such special instructions relating to interrupt and subroutine handling functions performed by the ICC. These instruction codes are all treated as non-operational instructions (NOPs) by the MicroController.

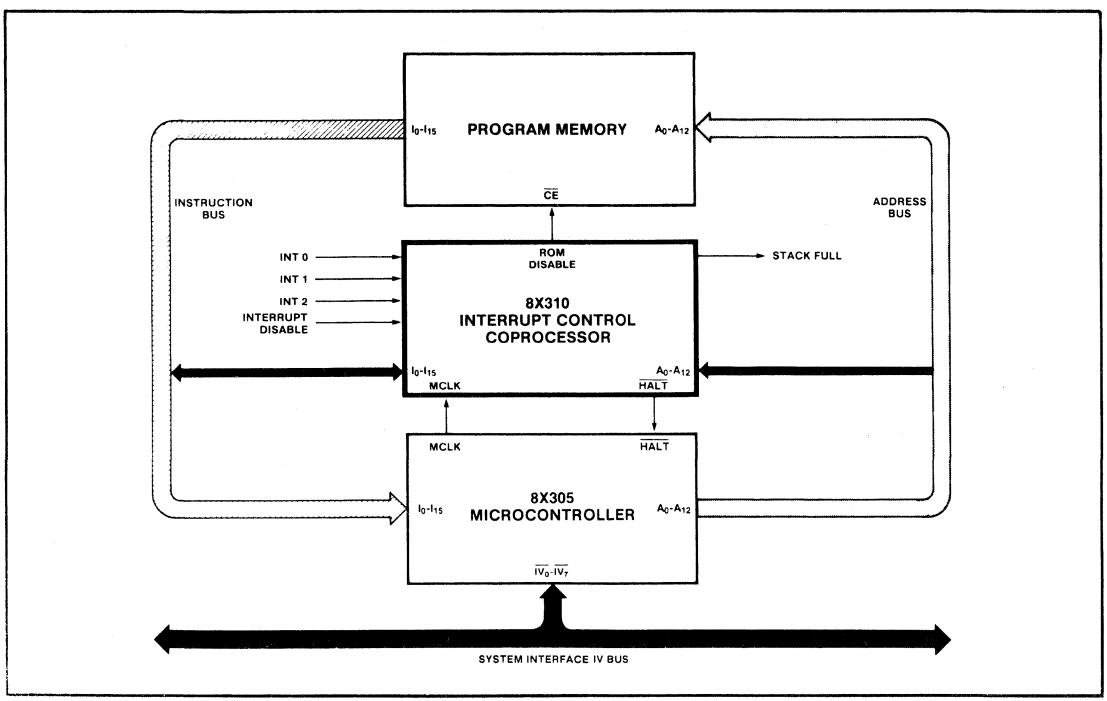


Figure 1. Typical System Connections Using ICC

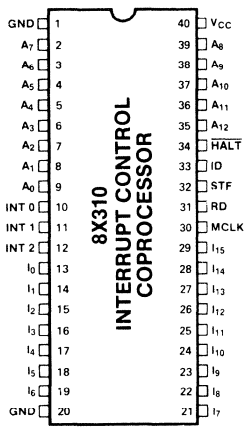
An internal one-bit mask is used to inhibit interrupt servicing. Whenever the mask is set, the ICC does not respond to any pending interrupt requests; however, any requests remain latched for future servicing. The mask can be set and cleared either by the user program or automatically during certain ICC functions. The special instructions SET MASK and CLEAR MASK are provided for user control. The Interrupt Disable input also inhibits interrupt request servicing.

The ICC provides a facility for implementing subroutines in the user program. A special PUSH instruction directs the ICC to store the return address into the stack in a manner similar to interrupt servicing. The jump to the subroutine, however, is performed by the user program. Subroutines may be nested (called from within other subroutines) depending on remaining vacancies in the four deep stack.

In general, the ICC adds some useful and very flexible facilities to the 8X305-based system. It offers both hardware and software capabilities that can improve efficiency and decrease program size. These features, from both a chip and system aspect, are described in subsequent paragraphs.

8X310 PACKAGE AND PIN DESIGNATIONS

N, I PACKAGE



TOP VIEW

ORDER NUMBERS

N8X310N, N8X310I
S8X310I/883B, S8X310I/883C

Pin No.	Identifier	Function
1, 20	GND	Ground. (Note: The printed circuit board should not use the ICC as a bridge for external ground.)
2-9, 35-39	A_7-A_0, A_{12}-A_8	Program address input lines from MicroController. Active high. A_0 is MSB.
10-12	INT 0-INT 2	Interrupt request input pins. INT 0 has the highest priority and INT 2 the lowest — Edge-triggered on a low-to-high transition.
13-19, 21-29	I_0-I_6, I_7-I_{15}	Bidirectional instruction bus; I_0 is MSB. When acting as an input, the ICC decodes the instruction flow (binary pattern on I_0-I_{15}) between program storage and the MicroController. During an interrupt or return cycle, the ICC outputs a JMP instruction to the MicroController via these lines —refer to FUNCTIONAL OPERATION of ICC.
30	MCLK	Master CLocK — active high input from 8X305 MicroController used for a timing reference and system synchronization.
31	RD	ROM (or PROM) Disable — active high output used to disable normal program storage so that the ICC can force an instruction to the MicroController.
32	STF	STack Full — active high output. When the LIFO stack is full, STF goes high and remains high until at least one register in the 4-level stack is empty.
33	ID	Interrupt Disable — active high. When this input pin is driven high, servicing of all interrupt requests is suspended.
34	$\overline{\text{HALT}}$	Active low output. Suspends all processing operations of the MicroController during period when the source of instruction data is changing between the ICC and program storage.
40	V_{CC}	+5 volt power supply.

FUNCTIONAL OPERATION

Basic Functions

The ICC performs the three general functions indicated below.

Function 1: Provides a means for the 8X305 MicroController to respond to interrupt requests by diverting the program flow of the 8X305 MicroController to the proper interrupt service routine or, in the case of a subroutine, the ICC stores the return address in the 4-level LIFO stack (Figure 2).

Function 2: Returns the user to the proper point in the main program for both interrupt and subroutine activities.

Function 3: Provides both automatic and programmed masking capabilities.

Interrupt Requests and Priority Considerations

An interrupt is requested when any one of the ICC input pins INT 0, INT 1, or INT 2 undergoes a low-to-high transition; this request is temporarily stored in an internal edge-triggered latch that corresponds to the affected interrupt input. The interrupt request latches are part of the Priority and Mask Logic shown in Figure 2. Unless masked or otherwise disabled, the ICC samples these latches once each instruction cycle. Any or all of the latches may be set when sampled by the ICC; however, only the interrupt of highest priority will be serviced — the remaining interrupts will be held in queue. Thus, if INT 0, INT 1 and INT 2 simultaneously compete for service, INT 0 is the first to be serviced followed, in order, by INT 1 and INT 2; likewise, if INT 1 and INT 2 compete for service, INT 1, being of higher priority will be serviced first. The CLEAR INTERRUPT instruction resets all interrupt request latches without affecting an interrupt service routine that is already in progress.

The highest priority interrupt request will be serviced when sampled by the ICC provided interrupts in general are not inhibited and a previous interrupt of equal or higher priority is not currently being serviced. The general masking of interrupts is discussed later. To determine priorities, the ICC keeps track of any interrupt that is serviced until the corresponding service routine returns. A subsequent interrupt request may interrupt a service routine in progress only if it is of a higher priority than that of the current interrupt being serviced. If, for example, INT 1 is requested and serviced, then before its service routine finishes, a request on INT 0 can be serviced as a second level interruption. However, a request on INT 2 or a second request on INT 1 must wait until the original INT 1 service routine returns. The interrupt service routine that was interrupted will resume execution at the point of interruption when the higher priority service routine returns (i.e. in the same manner as when returning to the main program).

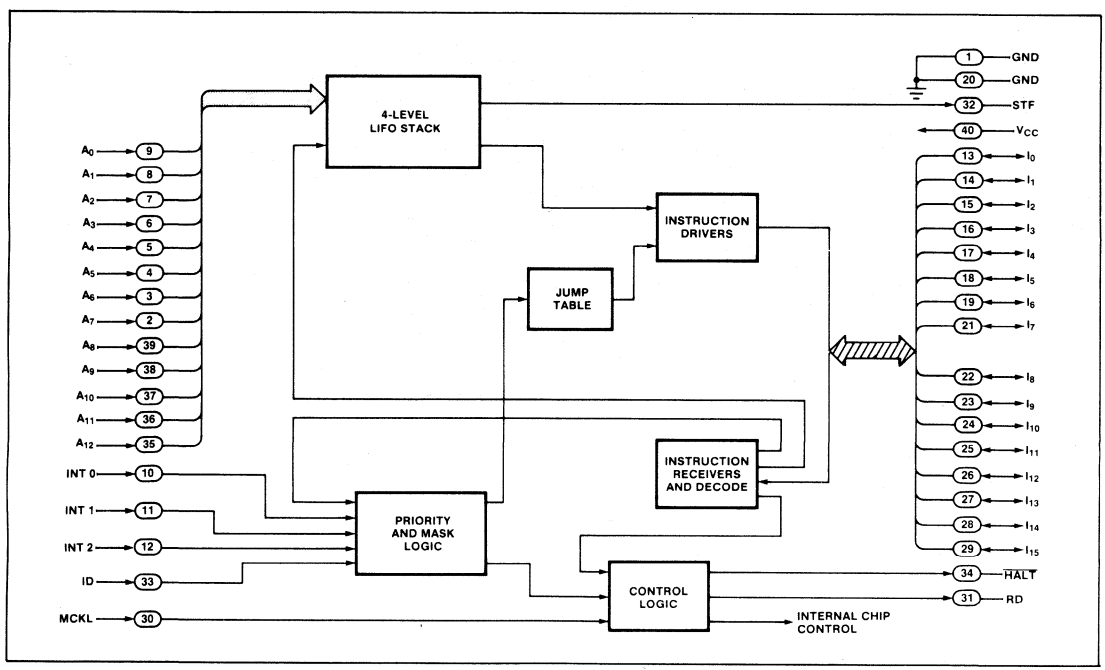


Figure 2. 8X310 Interrupt/Control Coprocessor — Functional Block Diagram

Interrupt Servicing

Interrupts are sampled only at the conclusion of an instruction cycle while the next instruction is being fetched from Program Memory.

When an interrupt request is serviced, the following general steps are performed:

- Address of the instruction that would normally be executed next is pushed into the 4-level LIFO Stack (Figure 2) for subsequent return to the main program.
- The ICC disables program storage and forces a JMP instruction onto the Instruction bus of the 8X305 MicroController. (Note: Because of timing considerations, the HALT signal is driven low to suspend operation of the MicroController for one instruction cycle; this permits the source of instruction data to change from program storage to the ICC without conflict.) The JMP instruction from the ICC transfers the MicroController to one of the three fixed program locations shown below. In each of these addresses, the user will normally store a JMP instruction to the interrupt service routine for that particular interrupt. Details of these operations are described later.

INT 0	Address 4
INT 1	Address 5
INT 2	Address 6

Return from Interrupt Service Routine

Upon completion of the interrupt service routine, the user codes the special RETURN instruction. When executed, the ICC performs the following steps

- The return address is popped from the LIFO stack.
- The ICC disables program storage and forces a JMP instruction onto the MicroController instruction bus with the return address from the stack. (The HALT signal is driven low for one instruction cycle.)
- The JMP instruction from the ICC transfers the MicroController to the instruction that was about to execute at the time the interrupt was taken

A typical structure for a user program which handles interrupts is shown in the following example:

Address	Instruction		Comment
0		(any)	First instruction executed after system reset
•		•	
•		•	
3		JMP MAIN	Jump around interrupt vector locations.
4		JMP SERV0	Service INT 0 interrupt.
5		JMP SERV1	Service INT 1 interrupt.
6		JMP SERV2	Service INT 2 interrupt.
7	MAIN	(any)	Continue main program.
•		•	
•		•	
	SERV0	(any)	Begin INT 0 service routine.
•		•	
•		•	
		MAIN R6,R6	ICC RETURN instruction. End INT 0 service routine (resume main program execution).
	SERV1	(any)	Begin INT 1 service routine.
•		•	
•		•	
		MOVE R6,R6	RETURN from INT 1 service routine.
	SERV2	(any)	Begin INT 2 service routine.
•		•	
•		•	
		MOVE R6,R6	RETURN from INT 2 service routine.

Instruction Set

The five instructions shown in Table 1 allow the user to efficiently manage both the interrupt and subroutine capabilities of the ICC in an 8X310/8X305-based system. When an ICC instruction appears in the program, it is interpreted as a NOP by the 8X305 but is captured and decoded by the ICC to perform the desired function. The capture-and-decode functions of the chip are automatic. Assembly and object codes for each ICC instruction are shown in Table 1.

Table 1. INSTRUCTION SET FOR ICC

Instruction	Instruction Codes		8X305 Operation	Description of ICC Operation
	Assembler	Binary		
		I_0 = MSB ↓ I_{15} = LSB ↓		
SET MASK	MOVE R5,R5	000 00101 000 00101	NOP	When executed, sets interrupt mask, thus inhibiting all interrupt servicing.
CLEAR MASK	MOVE R4,R4	000 00100 000 00100	NOP	When executed, clears interrupt mask for all interrupts
RETURN	MOVE R6,R6	000 00110 000 00110	NOP	When executed returns program to address at top of LIFO stack.
PUSH	MOVE R3,R3	000 00011 000 00011	NOP	Pushes "address + 2" onto stack if PUSH is programmed on odd address in program memory and "address + 1" if PUSH is programmed on even address.
CLEAR INTERRUPT	MOVE R2,R2	000 00010 000 00010	NOP	Clears all interrupt requests; an interrupt service routine that is in progress is unaffected.

Interrupt Masking Operations

Certain operations performed by the ICC and also some system considerations require that program execution not be interrupted for a specified interval of time. The servicing of interrupts by the ICC can be inhibited in a number of ways. Any time interrupts are inhibited, the ICC ignores any latched interrupt requests. However, the interrupt request latches are not cleared so that any previously pending requests remain latched. Also, during an interval when interrupt servicing is inhibited, any new interrupt signals received will get latched. As soon as interrupt servicing is enabled, any latched requests can be serviced on a priority basis.

The primary means of inhibiting interrupt servicing is the internal one-bit mask (latch). This mask can be set (to inhibit interrupts) or cleared under control of the user program using the special ICC instructions SET MASK and CLEAR MASK — See Table 1. With these instructions, segments of the user program can be isolated so as to proceed without interruptions. Frequently, uninterruptable segments are needed at the very beginning of the user program (initialization routine) and at the beginning of, or throughout an interrupt service routine. To facilitate this, the ICC automatically sets the mask whenever the MicroController executes address zero (typically resulting from a system reset) and whenever the ICC services an interrupt. The ICC also automatically clears the mask after performing a RETURN operation from an interrupt service routine; a RETURN from a subroutine does not affect the status of the interrupt mask.

The Interrupt Disable (ID) input pin may also be used to inhibit interrupt servicing. Interrupt servicing remains disabled as long as a high level is applied to the input. The ID input has no effect, however, on the status of the internal interrupt mask.

To ensure proper program flow, the ICC suspends interrupt servicing momentarily during certain situations. During the cycle in which the MicroController encounters an XEC (Execute) instruction an interrupt will not be serviced. This is because the XEC causes the MicroController to issue an address of an instruction to be executed out of the sequence of normal program flow. This would not be a valid address.

Interrupts are also suspended during execution of a PUSH or RETURN instruction and the instruction immediately following. This ensures proper operation of the LIFO stack. In addition, no interrupts are latched or serviced and no special ICC instructions are decoded at address zero which resets the ICC.

Subroutine Calling

The ICC provides for subroutine calling by storing the proper return address into the LIFO stack under control of the user program. Two instructions are required to implement a subroutine call — a PUSH instruction executed by the ICC and a JMP to the subroutine executed by the MicroController. The PUSH instruction is normally programmed at an odd-numbered address in program memory immediately followed by the JMP. When the PUSH instruction is executed, the ICC finds the address of the next instruction (JMP to subroutine) on the MicroController's address bus. internally changes the least-significant bit to one (effectively adds one to the address) and stores this into the stack. Program execution proceeds normally and the MicroController makes the jump to the beginning of the subroutine. The subroutine may be located at any convenient place in program memory.

Upon completion of the subroutine, the user codes the RETURN instruction in the same manner as for an interrupt service routine. At that point, the ICC forces the MicroController to resume execution of the main program at the instruction immediately following the JMP-to-subroutine instruction.

The code for a typical subroutine call-and-return is shown in the following example.

Address	Instruction	Comment
X (any odd-numbered address)	MOVE R3,R3	PUSH instruction initiates subroutine call by causing ICC to push the address X+2 onto the stack. (The PUSH instruction is interpreted as a NOP by the MicroController.)
X+1 (even)	JMP SUBR	The MicroController JMPs to the beginning of the subroutine.
X+2 (odd)	(any instruction)	Main program execution resumes here after RETURN from subroutine.
•	•	
•	•	
•	•	
SUBR (any address)	(any instruction)	Execution of subroutine starts here.
•	•	
•	•	
•	•	
(any address)	MOVE R6,R6	RETURN instruction causes ICC to transfer program back to X+2.

Stack Operation

The LIFO stack holds up to four 13-bit program addresses which allows the ICC to return from a subroutine or interrupt service routine. When all four stack locations are filled, the STack Full (STF) output pin is driven high and remains high until a RETURN (or reset) operation occurs. If an additional interrupt is serviced or subroutine called while the stack is full, the stack will overflow and the oldest return address will be overwritten and lost. That is, the stack retains the four most recent entries. After an overflow, the status of the STF output is not valid (until a reset operation occurs).

To prevent an interrupt from overflowing the stack, the user can connect the STF output directly to the Interrupt Disable (ID) input of the ICC. Then, even if the internal mask and priorities permit interrupt servicing, the interrupt request must still wait for the most recent service routine or subroutine to return.

Because subroutine calling is controlled explicitly by the user software, the user can always ensure that subroutine nesting alone could not overflow the stack. However, care must be taken whenever calling a subroutine from within an interrupt service routine since the number of remaining stack locations may vary at the time the interrupt is taken. If, for example, three stack locations are already filled (STF is low) at the time an interrupt is serviced, then a subroutine call executed within the interrupt service routine would cause the stack to overflow and the earliest return address to be lost.

As mentioned earlier, whenever a RETURN operation is performed from an interrupt service routine, the internal interrupt mask is automatically cleared. A RETURN from a

subroutine, however, does not affect the status of the mask. To accomplish this, a flag bit is added to each of the four stack locations which records whether each address pushed into the stack is caused by an interrupt or a subroutine call. This flag is read during a RETURN operation to determine whether or not the interrupt mask is cleared. This allows interrupt servicing and subroutine calls to be intermixed in any order.

Initialization

The ICC decodes address zero as a reset command to perform certain initialization functions. (Zero is the first address generated after the MicroController is reset.) Specifically, the Instruction-bus drivers are placed in a high-impedance state, HALT output is set high, RD output is set low, and all interrupt request latches are cleared. The interrupt mask is set so that any initialization routine by the user will not be interrupted until a CLEAR MASK instruction (MOVE R4,R4) is executed. The LIFO stack is reset to an empty state and the STF output is set low.

SYSTEM TIMING RELATIONSHIPS

Interrupt Servicing

Interrupt servicing begins at the end of a MicroController instruction cycle when the 8X305's MCLK signal goes from low-to-high. Starting from this point, processing of the interrupt proceeds as follows:

From the rising edge of MCLK 1:

- Interrupt mask is set to inhibit other interrupts.
- HALT output is driven low to stop internal operation of the 8X305 MicroController for one instruction cycle; MCLK is unaffected. ROM Disable (RD) output is driven high to disable program memory.

From the falling edge of MCLK 1:

- Takes address of next instruction from address bus and pushes it onto the top of stack to be used as the return address to the main program.

From the rising edge of MCLK 2:

- The ICC forces a JMP onto the instruction bus to one of three fixed vector addresses:

INT 0	Address 4
INT 1	Address 5
INT 2	Address 6

- Releases HALT (high) which allows the MicroController to complete the JMP to the above specified vector location in program memory.

From the rising edge of MCLK 3:

- Instruction-bus drivers of the ICC are disabled.
- ROM Disable (RD) pin is cleared (low) enabling the program memory which resumes control of the Instruction bus.

Return Operation

When the interrupt service routine or subroutine is completed, the RETURN instruction initiates the following sequence of events:

From the rising edge of MCLK 1:

- Interrupts are temporarily inhibited through third MCLK cycle.
- HALT output is driven low to stop MicroController for one instruction cycle.
- RD output is set high to disable program memory.

From the rising edge of MCLK 2:

- HALT output is driven high (cleared).
- A JMP instruction to address stored at top of LIFO stack is forced onto the Instruction bus by the ICC. (The stack is popped.)

From the rising edge of MCLK 3:

- Instruction-bus drivers of the ICC are disabled.
- RD is cleared enabling the program memory.
- If returning from an interrupt service routine (condition recorded in extra stack bit) the interrupt mask is cleared; otherwise the mask remains unaffected.

Once the preceding return actions are completed, the 8X305 MicroController will resume execution of the instruction at the return address.

APPLICATION HINTS

- When programming an interrupt service routine or subroutine, certain system operations typically need to be considered. In many interrupt-driven systems, a handshake signal is required to acknowledge the servicing of an interrupt request. The acknowledge signal may be transmitted by the interrupt service routine using a standard I/O port from the 8X300 Family.
- If the user wants to allow a higher priority interrupt request to interrupt a service routine, then the CLEAR MASK instruction should be programmed (perhaps after completing any critical operations).
- For both service routines and subroutines, the user may need to save the contents of some or all of the working registers of the MicroController so that operation of the main program is not upset. Registers may be written out to a working storage RAM such as 8X350 near the beginning of the routine, and restored from RAM just before returning to the main program.
- Certain subroutine calling techniques may be used to increase the efficiency of the user program. As shown in the following examples, a subroutine can automatically be repeated two, three or four times, if desired, without programming a loop.

SUBROUTINE AUTOMATICALLY EXECUTES TWICE

Address	Instruction		
X (even)	SUBR2	MOVE R3,R3	Push X+1 onto stack.
X+1 (odd)		(start of subroutine)	
•		•	
•		•	
		MOVE R6,R6	RETURN — First time jumps to X+1; second time jumps back to main program.

SUBROUTINE AUTOMATICALLY EXECUTES THREE TIMES

X (odd)	SUBR3	MOVE R3,R3	Push X+2 onto stack.
X+1 (even)		MOVE R3,R3	Push X+2 onto stack.
X+2 (odd)		(start of subroutine)	
•		•	
•		•	

SUBROUTINE AUTOMATICALLY EXECUTES FOUR TIMES

X (even)	SUBR4	MOVE R3,R3	Push X+1 onto stack.
X+1 (odd)		MOVE R3,R3	Push X+3 onto stack.
X+2 (even)		NOP	
X+3 (odd)		(start of subroutine)	
•		•	
•		•	

- In a manner similar to the MicroController multi-way branch technique, one of several subroutines can be selected according to an index value.

SUBROUTINE CALL SELECTED BY VALUE IN R1

Address	Instruction		
X (odd)		MOVE R3,R3	Push X+2 onto stack.
X+1 (even)		XEC TABLE (R1)	Execute JMP at TABLE + (R1)
X+2 (odd)		(any)	Subroutine returns here.
•		•	
•		•	
(any)	TABLE	JMP SUB0	Call SUB0 if R1 = 0.
		JMP SUB1	Call SUB1 if R1 = 1.
		•	
		•	

DC ELECTRICAL CHARACTERISTICS

COMMERCIAL: V_{CC} = 5.0 V (±5%); 0° C ≤ T_A ≤ 70° C
MILITARY: V_{CC} = 5.0 V (±10%); T_A ≥ -55° C
T_C ≤ 125° C

ABSOLUTE MAXIMUM RATINGS

Parameter		Rating	Unit	Parameter		Rating	Unit
V_{CC}	Power supply voltage	+7	V DC	V_O	Off-state output voltage	+5.5	V DC
V_{IN}	Input voltage	+5.5	V DC	T_{STG}	Storage temperature range	−65 to +150	°C

Parameter		Test Conditions		Limits (Commercial) Min	Typ	Max	Limits (Military) Min	Typ	Max	Unit
V_{IH}	High Level Input Voltage			2.0			2.0			V
V_{IL}	Low Level Input Voltage					0.8			0.8	V
V_{OH}	High Level Output Voltage	V_{CC} = Min.; I_{OH} = −1.0 mA		2.4			2.4			V
V_{OL}	Low Level Output Voltage	V_{CC} = Min.	COMMERCIAL: I_{OL} = 8 mA			0.55				V
			MILITARY: I_{OL} = 4.25 mA						0.55	
V_{CL}	Input Clamp-Diode Voltage	V_{CC} = Min; I_{CL} = −10 mA				−1.5			−1.5	V
I_{IH}	High Level Input Current	V_{CC} = Max; V_{IH} = 2.7 V				100			100	μA
I_{IL}	Low Level Input Current	V_{CC} = Max; V_{IL} = 0.4 V				−550			−700	μA
I_{OS}	Short Circuit Output Current	V_{CC} = Max; V_O = 0 V		−15		−80	−15		−80	mA
I_{CC}	Supply Current	V_{CC} = Max; I_0-I_{15} = High-Z								mA
			T_A = 0° C[2]			200				
			T_A = 70° C			185				
			T_A = −55° C[2]						230	
			T_C = 125° C						170	

AC ELECTRICAL CHARACTERISTICS

LOADING: See TEST LOADING CIRCUITS

COMMERCIAL: V_{CC} = 5.0 V (±5%); 0° C ≤ T_A ≤ 70° C
MILITARY: V_{CC} = 5.0 V (±10%); T_A ≥ -55° C
T_C ≤ 125° C

Parameter		References From	To	Test Conditions	Limits (Commercial) Min	Typ	Max	Limits (Military) Min	Typ	Max	Unit
Pulse Widths:											
t_{WIH}	Interrupt High	↑INT_i	↓INT_i		30			30			ns
t_{WIL}	Interrupt Low	↓INT_i	↑INT_i		35			35			ns
t_{WMH}	MCLK High	↑MCLK	↓MCLK	For all Functions	40			47			ns
Propagation Delays:											
t_{PRH}	RD High	↑MCLK	↑RD	Interrupt or Return			70			75	ns
t_{PRL}	RD Low	↑MCLK	↓RD	Interrupt or Return			15			17	ns
t_{PHL}	HALT Low	↑MCLK	↓HALT	Interrupt or Return			70			87	ns
t_{PHH}	HALT High	↑MCLK	↑HALT	Interrupt or Return			65			75	ns
t_{PSH}	Stack Full High	↓MCLK	↑STF	Interrupt or Subroutine Call			105			105	ns
t_{PSL}	Stack Full Low	↓MCLK	↓STF	Return or Reset			110			115	ns

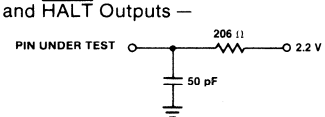
AC ELECTRICAL CHARACTERISTICS (CONTINUED)

Parameter		References From	References To	Test Conditions	Limits (Commercial) Min	Typ	Max	Limits (Military) Min	Typ	Max	Unit
Setup Times:											
t_{SIH}	Interrupt Input Setup[3]	↑INT_i	↑MCLK		35			35			ns
t_{SA}	Address Setup	A_0-A_{13}	↑MCLK	Interrupt, Subroutine Call, or Reset	0			0			ns
t_{SC}	Instruction Setup[5]	I_{0-15}	↓MCLK	All Commands	Note 5			Note 5			ns
t_{SD}	Interrupt Disable Setup[3]	ID	↑MCLK		30			30			ns
Hold and Reset Recovery Times:											
t_{HIL}	Interrupt Low Input Hold[3]	↑MCLK	↑INT_i		15			15			ns
t_{HA}	Address Hold	↓MCLK	A_{0-13}	Subroutine Call or Reset	75			90			ns
t_{HC}	Instruction Hold	↓MCLK	I_{0-15}	All Commands	55			55			ns
t_{HD}	Interrupt Disable Hold[3]	↑MCLK	ID		25			25			ns
t_{RI}	Interrupt Reset Recovery[4]	↓MCLK	↑INT_i	Reset or Cancel Command	70			70			ns
Output Enable/Disable Delays:											
t_{OEC}	Instruction Output Enable	↑MCLK	I_{0-15}	Interrupt or Return			70			87	ns
t_{ODC}	Instruction Output Disable	↑MCLK	I_{0-15}	Interrupt or Return			40			47	ns

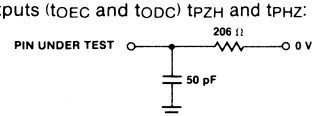
Notes:
1. All electrical characteristics are guaranteed after power is applied and thermal equilibrium has been reached.
2. The 200 and 230 milliampere values are worst case over the entire temperature range for the Commercial and Military parts, respectively.
3. Parameters t_{SIH}, t_{HIL}, T_{SD}, and t_{HD} are used only to determine whether an interrupt request will be serviced during the current or a subsequent instruction cycle. The INT_i and ID inputs are asynchronous and transitions on either input may safely occur at any time with respect to MCLK. A low-to-high transition on INT_i occurring after t_{SIH} and before t_{HIL} means only that it cannot be determined for sure whether or not the interrupt request will be honored during the current instruction cycle. Similarly, transitions on ID between t_{SD} and t_{HD} make it uncertain as to whether or not masking applies during the current instruction cycle.
4. When clearing interrupt requests (including a reset operation), any new low-to-high transitions appearing at the INT_i inputs that occur before t_{RI} risk being cleared and therefore ignored; however, any transition after t_{RI} is certain to be latched.
5. COMMERCIAL: t_{SC} (minimum) = 15 ns — t_{PRL} (actual).
   MILITARY: t_{SC} (minimum) = 17 ns — t_{PRL} (actual).
   (The required instruction enable time for the program memory depends on the sum of the t_{PRL} and t_{SC}.)

TEST SETUPS

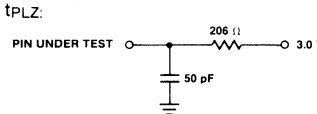
RD, STF, and $\overline{HALT}$ Outputs —



I_0-I_{15} Outputs (t_{OEC} and t_{ODC}) t_{PZH} and t_{PHZ}:



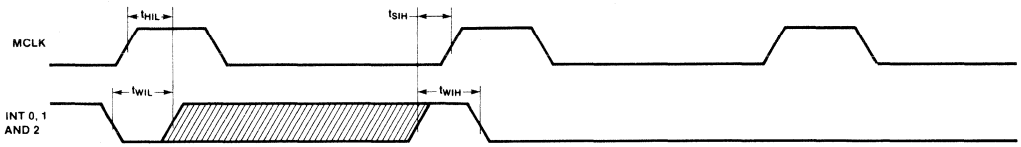
t_{PZL} and t_{PLZ}:



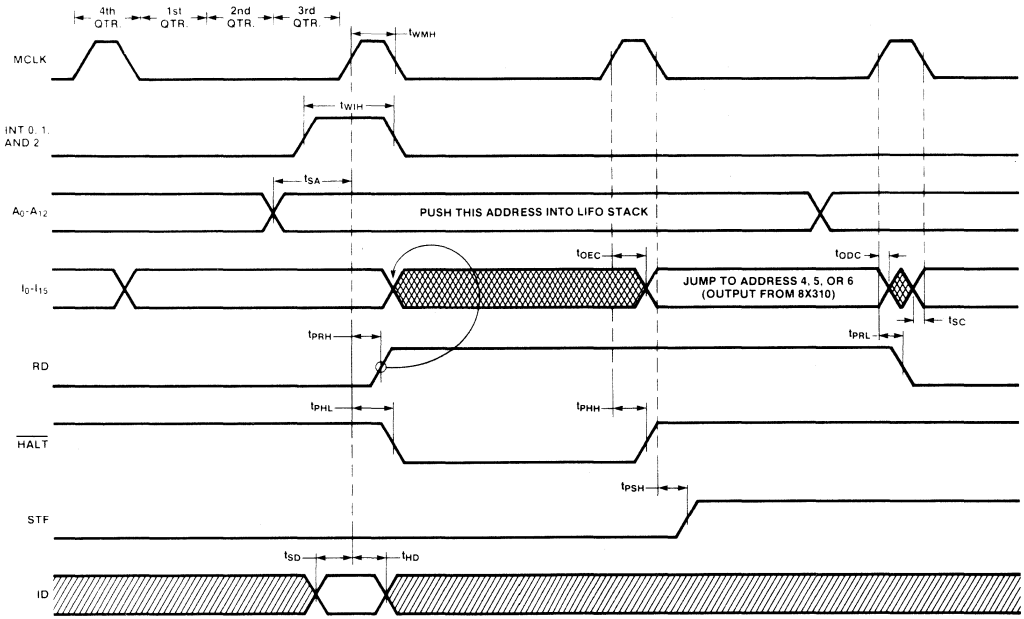
Notes:
1. V_{MEAS} = 1.5 V for all input signals and RD, STF, and HALT outputs.
2. For I_0-I_{15} (t_{OEC}): V_{MEAS} = 1.5 V
   For I_0-I_{15} (t_{ODC}): V_{MEAS} (t_{PLZ}) = V_{OL} + 0.5 V
   V_{MEAS} (t_{PHZ}) = V_{OH} − 0.5 V

TIMING DIAGRAMS

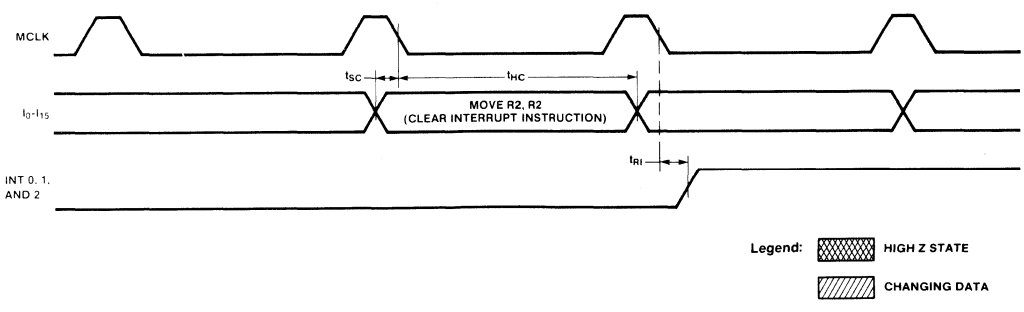
INTERRUPT REQUEST TIMING:



INTERRUPT SERVICE TIMING:

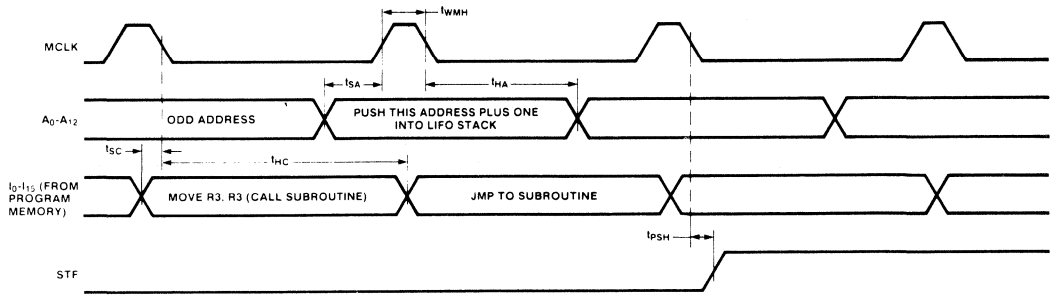


CLEAR INTERRUPT INSTRUCTION TIMING:

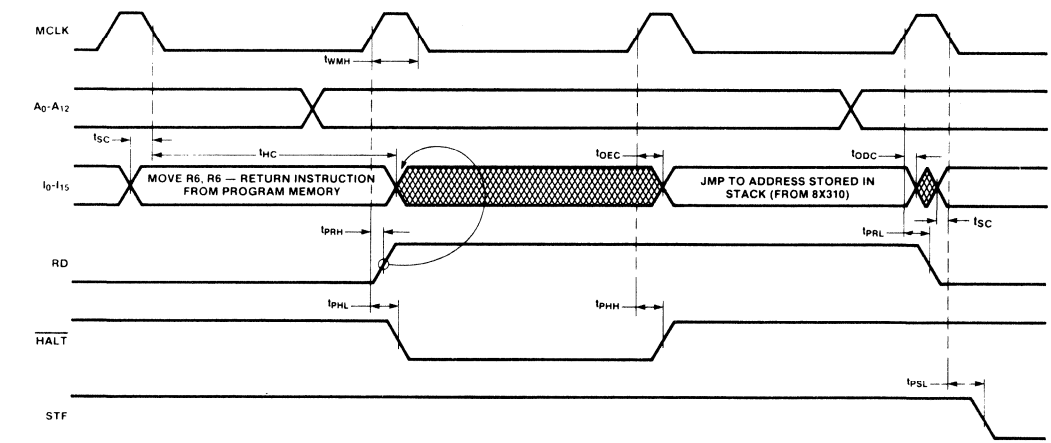


TIMING DIAGRAMS (Continued)

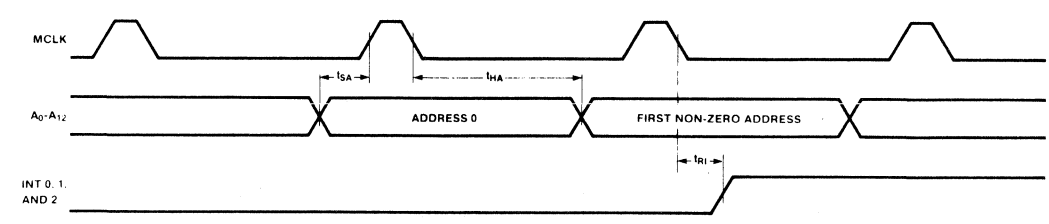
CALL SUBROUTINE TIMING:



RETURN TIMING (FROM INTERRUPT OR SUBROUTINE):



RESET TIMING:



Legend: HIGH Z STATE

BUS INTERFACE REGISTER ARRAY

Originally published by Signetics January 1984

FEATURES

- **16-byte/2-port interface**
- **8- or 16-bit primary port (Host) interface (User selectable)**
- **8-bit secondary port interface**
- **Two 8-bit flag registers (handshake control)**
- **DMA or programmed I/O operation**
- **Two three-state bidirectional ports**
- **Secondary port is bus compatible with 8X305**
- **Single 5V supply**
- **40-pin package**

ARCHITECTURAL OVERVIEW

The Signetics 8X320 Bus Interface Register Array (Figure 1) is a dual-port RAM memory designed for use between a host processor and a peripheral processor. Specifically, the register array provides a convenient and economical interface between the 8X305 (or 8X300) Microcontroller (secondary port) and User's Host System (primary port); the host can be almost any bus-oriented device—another processor, a minicomputer, or a mainframe computer. The host has 8-bit (byte) or 16-bit (word) access to the primary port; data can be read-from or written-into any memory location as determined by the primary-port address and control lines. The secondary port (8X305 bus) consists of eight input/output lines and four bus control lines. To implement the secondary-port interface, an 8-bit memory location is addressed during one machine cycle and, during another cycle, data is read or written under control of the secondary (8X305) processor. Both primary and secondary ports feature three-state outputs and both ports are bidirectional.

Besides the convenience and economy of a two-port memory, the array also provides simple handshake control via two 8-bit flag registers, logic to facilitate DMA transfers, and a write-protect feature for the primary port in both byte and word modes of operation.

BLOCK DIAGRAM

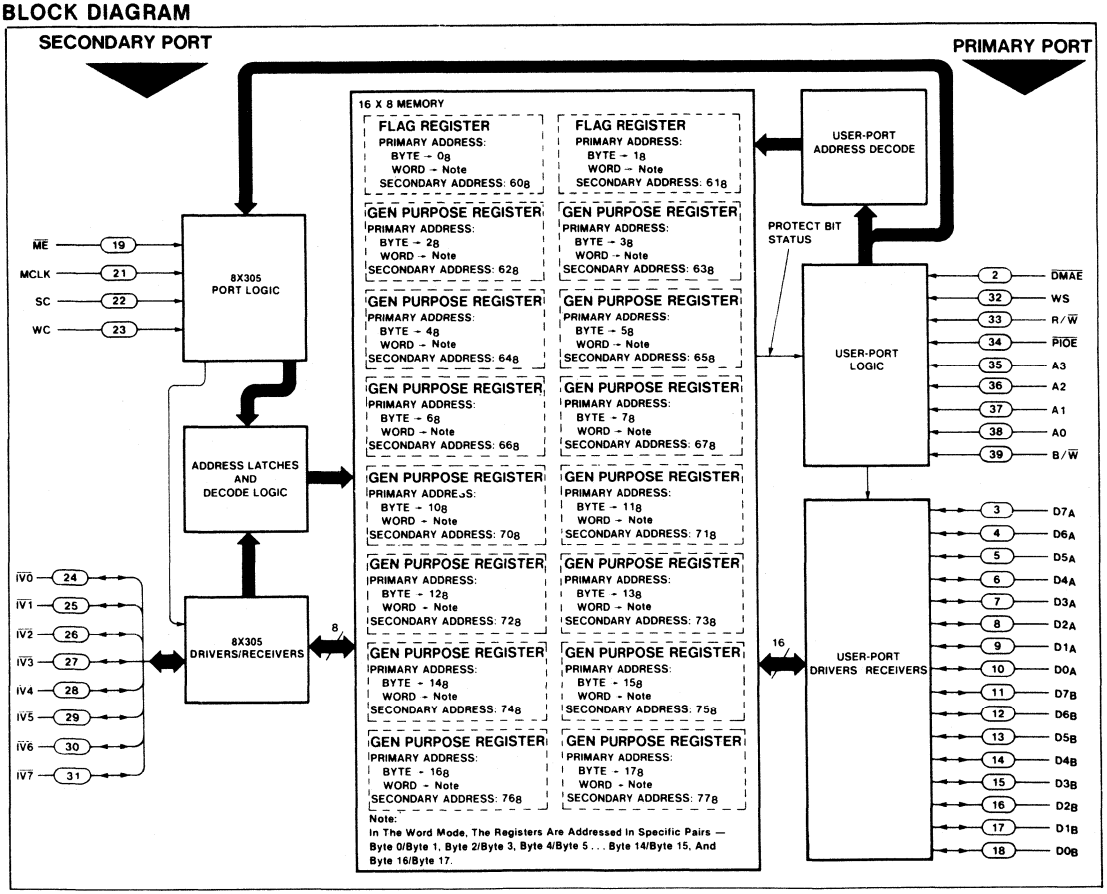
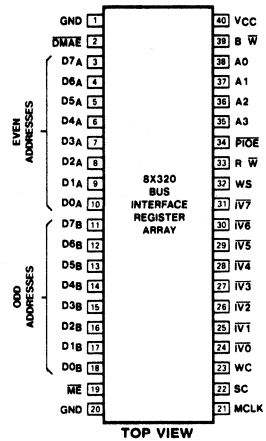


Figure 1. Block Diagram of 8X320 Bus Interface Register Array

N, I PACKAGE



TOP VIEW

ORDER NUMBERS

N8X320N, N8X320I

PIN NO.	PARAMETER		FUNCTION
1, 20	GND	Ground	Circuit ground.
2	$\overline{DMAE}$	Direct Memory Access Enable	Enables primary port to facilitate DMA transfers; does not affect secondary port.
3-18	$D0_A$-$D7_A$/ $D0_B$-$D7_B$	Primary Data Port	Sixteen 3-state lines used for data transfers to-and-from the primary data port; most significant bit is $D0_B$ and least significant bit is $D7_A$.
19	$\overline{ME}$	Master Enable	Enables secondary port when active low ($\overline{ME}$).
21	MCLK	Master Clock	When MCLK is high, and 8X320 is enabled ($\overline{ME}$ = Low), a register location may be either selected or written-into under control of SC and WC.
22	SC	Select Command	With SC high, WC low, MCLK high and $\overline{ME}$ low, data on $\overline{IV0}$ through $\overline{IV7}$ is interpreted as an address. If any one of the 16 register addresses (60_B-77_B) matches that on the I/O (IV) bus, that particular register is selected and remains selected until another address on the same bank (i.e. $\overline{ME}$ = low) is output on the I/O bus—at which time, the old register is deselected and a new register may or may not be selected.
23	WC	Write Command	With WC high, SC low, MCLK high, and $\overline{ME}$ low, the selected register stores contents of $\overline{IV0}$-$\overline{IV7}$ as data.
24-31	$\overline{IV0}$-$\overline{IV7}$	Secondary Data Port	Eight 3-state lines used to transfer data or I/O address to-and-from the secondary data port; most significant bit is $\overline{IV0}$ and least significant bit is $\overline{IV7}$.
32	$\overline{WS}$	Write Strobe	When active high, data appearing at the primary port ($D0_A$-$D7_A$/$D0_B$-$D7_B$) is stored in the register array if the primary port is in the *write* mode.
33	R/$\overline{W}$	Read/Write Control	When this signal is high, primary port is in *read* mode; when signal is low, primary port is in *write* mode.
34	$\overline{PIOE}$	Programmed I/O Enable	When active low, primary port operates in programmed input/output mode with register to be read-from or written-into selected by A0-A3.
35-38	A0-A3	Primary Port Address Select	Selects register or register-pair that primary port is to read-from or write-into. Most significant bit is A3; least significant bit is A0.
39	B/$\overline{W}$	Byte/Word	When signal is high, the primary port operates in the byte (8-bit) mode; when signal is low, the primary port operates in the word (16-bit) mode.
40	V_{CC}	Power	+5 volts.

All barred symbols ($\overline{DMAE}$, etc.) denote signals that are asserted (or active) when low (logical 0); signals that are not barred are asserted in the high state (logical 1).

OPERATING CHARACTERISTICS

Memory Organization

Memory and address correlation for the 16-register array is shown in Figure 2. From the primary port, the sixteen 8-bit registers can be addressed in either (8-bit) or word (16-bit) format; in the word mode, the registers are addressed in pairs—$0_8/1_8$, $2_8/3_8$, $4_8/5_8$, . . . $14_8/15_8$, and $16_8/17_8$. From the secondary port, all registers are addressed in byte format—60_8 through 77_8. The memory consists of two 8-bit flag registers and fourteen 8-bit general-purpose registers. The flag registers facilitate information transfers between the two ports and, in addition, they protect certain registers from being written into from the primary port.

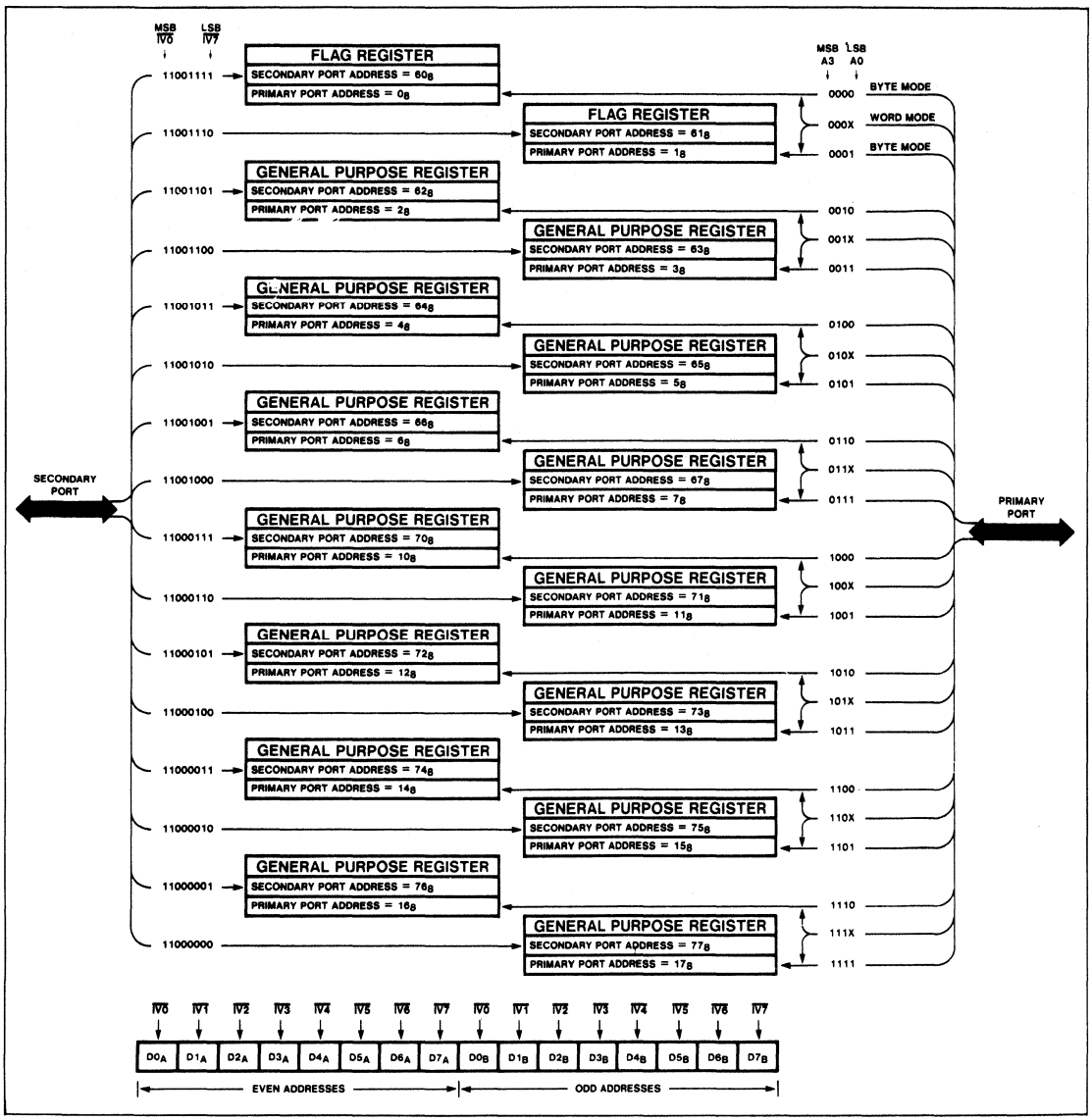


Figure 2. Memory and Address Organization for the 8X320

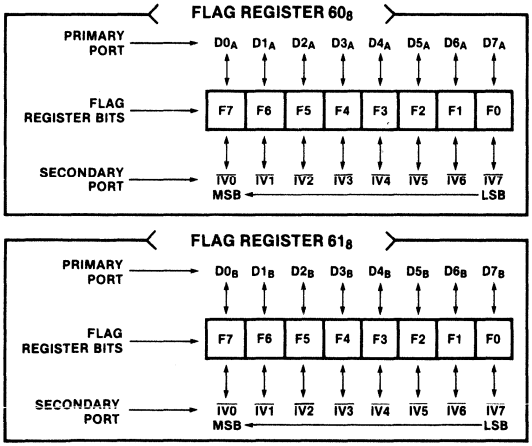
In either byte or word mode, the write-protect logic, implemented by bits F0 and F1 of register 60_8, inhibits the primary port from writing into addresses 16_8 and 17_8, respectively. Both write-protect bits (F0 and F1) can be read or written from the secondary port; the bits are read-only from the primary port.

As shown in Table 1, flag bits F2 through F7 of 60_8 and F0 through F7 of 61_8 are controlled by the fourteen general-purpose registers. When any one of these registers is written into by either port, the corresponding flag bit for that register is automatically set by internal logic of the 8x320. When information is read from any register, the corresponding flag bit must be reset by user software. Except for the write-protect bits, all other flag bits can be read or reset from the primary or the secondary port. Table 2 shows the relationship between bits of the flag registers and bits of the primary and secondary ports.

Table 1. CONTROL OF THE TWO FLAG REGISTERS

Flag Registers	60_8 (0_8)	F2 F3 F4 F5 F6 F7
	61_8 (1_8)	F0 F1 F2 F3 F4 F5 F6 F7
Octal Address of Controlling Byte	Primary	2 3 4 5 6 7 10 11 12 13 14 15 16 17
	Secondary	62 63 64 65 66 67 70 71 72 73 74 75 76 77

Table 2. RELATIONSHIP BETWEEN FLAG REGISTER BITS AND THOSE OF PRIMARY AND SECONDARY PORTS



FUNCTION AND CONTROL OF PRIMARY PORT

The primary port provides an 8-bit (byte) or 16-bit (word) interface between the 16-byte memory and the user's host system. If the host is an 8-bit system (or 16-bit system operating in Byte mode), the sixteen bidirectional I/O lines must be tied together ($D0_A$ to $D0_B$, $D1_A$ to $D1_B$, . . . and $D7_A$ to $D7_B$); when data is input or output on $D0_A$ through $D7_A$, the remaining eight lines ($D0_B$ through $D7_B$) are high-Z and vice-versa.

Other than the Byte/$\overline{\text{Word}}$ control line, specific operating characteristics of the primary port are controlled by two signals—$\overline{\text{PIOE}}$ (Programmed I/O Enable) and $\overline{\text{DMAE}}$ (Direct Memory Access Enable). When $\overline{\text{PIOE}}$ is active (low) and $\overline{\text{DMAE}}$ is inactive (high), the primary port operates in the programmed I/O mode—refer to Table 3; in this mode of operation, the register to be read-from or written-into is determined by four address lines (A0 through A3) and the Byte/$\overline{\text{Word}}$ control line—see Figure 2 and Table 4. In the DMA mode of operation, A1, A2, and A3 are not used; data is read-from or written-into preassigned registers: bytes 16_8 (76_8) and 17_8 (77_8) for the byte mode of operation and bytes 14_8 (74_8)/15_8 (75_8) and 16_8 (76_8)/17_8 (77_8) for the word mode of operation. In both cases, switching between bytes 16_8 and 17_8 in the byte mode and 14_8/15_8 and 16_8/17_8 in the word mode is controlled by A0 (the least significant address bit). Refer to Table 5.

Table 3. MODE CONTROL OF PRIMARY PORT

MODE	$\overline{\text{PIOE}}$	$\overline{\text{DMAE}}$
Disabled (output)	1	1
Programmed I/O	0	1
DMA	X	0

X = Don't Care

Table 4 defines programmed I/O operation of the primary port in terms of read/$\overline{\text{write}}$ functions and Byte/$\overline{\text{Word}}$ control. In the byte mode, data is read-from or written-into the even addresses (0_8, 2_8, 4_8, 6_8, 10_8, 12_8, 14_8, and 16_8) via data lines $D0_A$ through $D7_A$; data is read-from or written-into odd addresses (1_8, 3_8, 5_8, 7_8, 11_8, 13_8, 15_8, and 17_8) via data lines $D0_B$ through $D7_B$. When A0 is low (logical 0), even addresses are selected and when A0 is high (logical 1), odd addresses are selected; thus, A0 is the LSB of a 4-bit address. In the word mode, the state of A0 is irrelevant, since both the odd and even bytes are, simultaneously, read-from or written-into; thus, a register pair is selected by a 3-bit address, A1 being the LSB.

In the DMA mode of operation with $\overline{\text{DMAE}}$ set to 0 and other conditions satisfied, data is directly transferred to-or-from specified memory locations under control of Byte/$\overline{\text{Word}}$, R/$\overline{\text{W}}$, and A0. The state of the Byte/$\overline{\text{Word}}$ control line determines whether the data word is 8 bits or 16 bits. The A0 address line correlates eight of

Table 4. PRIMARY PORT OPERATING IN PROGRAMMED I/O MODE

MODE	B/$\overline{\text{W}}$	A0	$D0_A$–$D7_A$ (Even Addresses)	$D0_B$–$D7_B$ (Odd Addresses)
Read	0 (Word)	X	Stored Data	Stored Data
Read	1 (Byte)	0	Stored Data	HI-Z
Read	1 (Byte)	1	HI-Z	Stored Data
Write	0 (Word)	X	Write	Write
Write	1 (Byte)	0	Write	No Change
Write	1 (Byte)	1	No Change	Write

X = Don't Care

the sixteen data lines ($D0_A$-$D7_A$ or $D0_B$-$D7_B$) with the proper byte/word location. Thus, in the word mode, the exchange of data between the memory and the primary port occurs via $D0_A$-$D7_A$ for bytes 14_8 and 16_8 and via $D0_B$-$D7_B$ for bytes 15_8 and 17_8. The byte mode of operation is similar, except that the unused eight lines are three-stated.

FUNCTION AND CONTROL OF SECONDARY PORT

The secondary port provides an 8-bit interface between the sixteen memory registers and the 8X305 (or other processor). As shown in Table 6, the secondary-port interface is controlled by five input signals and a status latch. The status latch is set when SC is high (MCLK high/$\overline{ME}$ low) and a valid memory address (60_8-77_8) is presented to the 8X320 via the secondary data port ($\overline{IV0}$-$\overline{IV7}$). The latch is cleared by internal logic when an invalid memory address is presented at the secondary port. In all read/write operations from the secondary port, the status latch acts like a master enable; data can be transferred only if the status latch is set.

Table 5. DMA OPERATION OF THE PRIMARY PORT

MODE	BYTE/WORD	A0	$D0_A$-$D7_A$	$D0_B$-$D7_B$
Read	0 (Word)	0	Data stored in byte 14_8	Data stored in byte 15_8
Read	0 (Word)	1	Data stored in byte 16_8	Data stored in byte 17_8
Read	1 (Byte)	0	Data stored in byte 16_8	HI-Z
Read	1 (Byte)	1	HI-Z	Data stored in byte 17_8
Write	0 (Word)	0	Write to byte 14_8	Write to byte 15_8
Write	0 (Word)	1	Write to byte 16_8	Write to byte 17_8
Write	1 (Byte)	0	Write to byte 16_8	HI-Z
Write	1 (Byte)	1	HI-Z	Write to byte 17_8

Table 6. FUNCTIONAL CONTROL OF SECONDARY PORT

$\overline{ME}$	SC[1]	WC[1]	MCLK	R/$\overline{W}$	STATUS LATCH	FUNCTION OF SECONDARY BUS
L	L	L	X	X	Set	Output data from 8X320 memory to 8X305
L	L	H	H	H	Set	Data from 8X305 is input and written-into a previously-selected memory location of the 8X320 (Note 2).
L	L	H	H	L	Set	With the primary port in the write mode (R/$\overline{W}$ = 0), the secondary port is overridden and cannot write to the same register addressed by the primary port; however, the register addressed by the primary port can be read and any other register can be read-from or written-into from the secondary port (Note 2).
L	H	L	H	X	X	Data transmitted to the secondary port via the $\overline{IV}$ bus is interpreted as an address; if address is within range of 60_8-77_8 the memory status latch is subsequently set.
L	L	H	L	X	X	Inactive
L	H	L	L	X	X	Inactive
L	L	X	X	X	Not Set	Inactive
H	X	X	X	X	X	Inactive

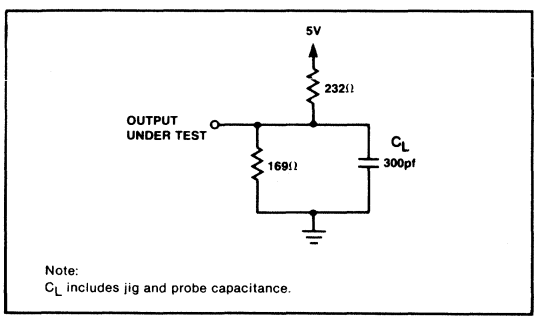
Notes:
1. The SC and WC lines should never both be high at the same time; the 8X305 processor never generates this condition.
2. During read or write operations, the same register can be simultaneously addressed from either port. For any write operation by both ports on the same register, the primary port has priority; other than this, the 8X320 does not indicate error conditions or resolve conflicts.
3. X = Don't Care.

DC CHARACTERISTICS $0°C \leq T_A \leq 70°C$; $4.75V \leq V_{CC} \leq 5.25V$

PARAMETER		TEST CONDITIONS[1, 2]	LIMITS			UNIT
			Min	Typ	Max	
V_{CC}	Supply voltage		4.75	5	5.25	V
$V_{IN\ (L)}$	Low level input voltage				0.80	V
$V_{IN\ (H)}$	High level input voltage		2.0			V
V_{OL}	Low level output voltage	V_{CC} = 4.75V; I_{OL} = 16mA			0.55	V
V_{OH}	High level output voltage	V_{CC} = 4.75V; I_{OH} = −3mA	2.40			V
V_{CL}	Input clamp voltage	I_I = −5mA			−1.00	V
I_{CC}	Supply current	V_{CC} = 5.25V (Both ports high-Z)			270	mA
I_{OS}	Short circuit output current[3]	V_{CC} = 4.75V	−20		−100	mA
$I_{IN\ (L)}$	WC, MCLK, SC, & $\overline{ME}$	V_{CC} = 5.25V; V_{IL} = 0.50V			−1.0	mA
$I_{IN\ (L)}$	B/$\overline{W}$	V_{CC} = 5.25V; V_{IL} = 0.50V			−1.6	mA
$I_{IN\ (L)}$	A0-A3	V_{CC} = 5.25V; V_{IL} = 0.50V			−1.0	mA
$I_{IN\ (L)}$	$\overline{DMAE}$	V_{CC} = 5.25V; V_{IL} = 0.5V			−800	μA
$I_{IN\ (L)}$	WS, $\overline{PIOE}$, & R/$\overline{W}$	V_{CC} = 5.25V; V_{IL} = 0.5V			−400	μA
$I_{IN\ (L)}$	$\overline{IV0}$-$\overline{IV7}$	V_{CC} = 5.25V; V_{IL} = 0.5V			−400 each line	μA
$I_{IN\ (L)}$	$D0_A$-$D7_A$/$D0_B$-$D7_B$	V_{CC} = 5.25V; V_{IL} = 0.5V			−400 each line	μA
$I_{IN\ (H)}$	WC, SC, MCLK, & $\overline{ME}$	V_{CC} = 5.25V; V_{IH} = 5.25V			100	μA
$I_{IN\ (H)}$	B/$\overline{W}$	V_{CC} = 5.25V; V_{IH} = 5.25V			240	μA
$I_{IN\ (H)}$	A0	V_{CC} = 5.25V; V_{IH} = 5.25V			120	μA
$I_{IN\ (H)}$	A1-A3	V_{CC} = 5.25V; V_{IH} = 5.25V			60	μA
$I_{IN\ (H)}$	$\overline{DMAE}$	V_{CC} = 5.25V; V_{IH} = 5.25V			120	μA
$I_{IN\ (H)}$	WS, $\overline{PIOE}$, & R/$\overline{W}$	V_{CC} = 5.25V; V_{IH} = 5.25V			60	μA
$I_{IN\ (H)}$	$\overline{IV0}$-$\overline{IV7}$ and $D0_A$-$D7_A$/$D0_B$-$D7_B$	V_{CC} = 5.25V; V_{IH} = 5.25V			100	μA

Notes:
1. Operating temperature ranges are guaranteed after terminal equilibrium has been reached.
2. All voltages are measured with respect to ground terminal.
3. Short only one output at a time.

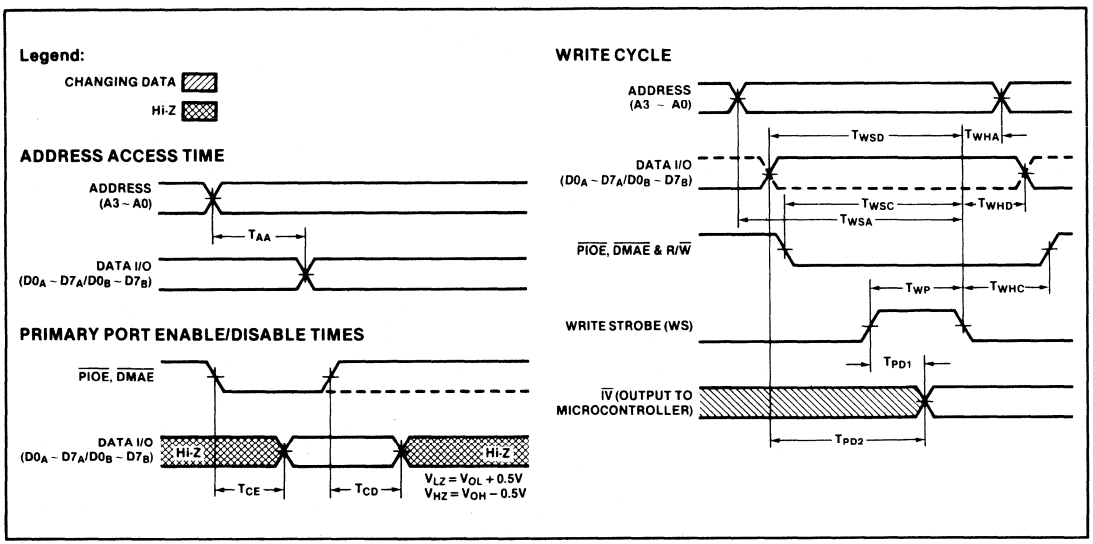
TEST CIRCUIT



Note:
C_L includes jig and probe capacitance.

AC CHARACTERISTICS OF PRIMARY PORT

$0°C \leq T_A \leq 70°C$; $4.75V \leq V_{CC} \leq 5.25V$
Loading: See Test Circuit

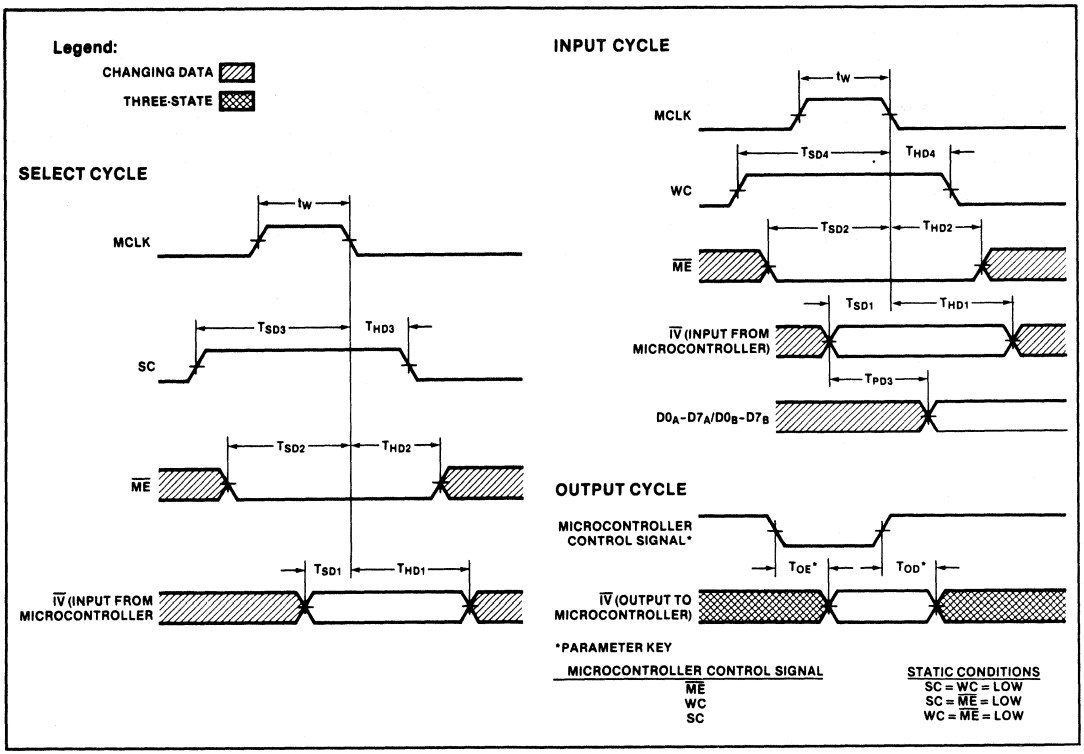


	PARAMETER	FROM	TO	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT
T_{AA}	Address Access Time	A3-A0	$D0_A$-$D7_A$/$D0_B$-$D7_B$			45	ns
T_{CE}	Primary port enable time	↓PIOE ↓DMAE	$D0_A$-$D7_A$/$D0_B$-$D7_B$			30	ns
T_{CD}	Primary port disable time	↑PIOE ↑DMAE	$D0_A$-$D7_A$/$D0_B$-$D7_B$			35	ns
T_{WSA}	Address setup time	A3-A0	↓WS	40			ns
T_{WHA}	Address hold time	↓WS	A3-A0	0			ns
T_{WSD}	Primary port data setup time	$D0_A$-$D7_A$/$D0_B$-$D7_B$	↓WS	30			ns
T_{WHD}	Primary port data hold time	↓WS	$D0_A$-$D7_A$/$D0_B$-$D7_B$	0			ns
T_{WSC}	Write mode control setup time	PIOE DMAE R/W	↓WS ↓WS ↓WS	30 40 30			ns ns ns
T_{WHC}	Write mode control hold time	↓WS	PIOE DMAE R/W	10 10 10			ns ns ns
T_{WP}	Write strobe pulse width			25			ns
T_{PD1}[1]	Primary port data delay	$D0_A$-$D7_A$/$D0_B$-$D7_B$	IV0-IV7			75	ns
T_{PD2}[2]	Primary port data delay from WS	↑WS	IV0-IV7			75	ns

Notes:
1. Measurement with Write Strobe set High and the control signals of the secondary port set for output data from the same register.
2. Measurement with primary port data stable and control signals of secondary port set for output data from the same register.

AC CHARACTERISTICS OF SECONDARY PORT

$0°C \leq T_A \leq 70°C$, $4.75V \leq V_{CC} \leq 5.25V$
Loading: See Test Circuit



	PARAMETER	FROM	TO	LIMITS			UNIT
				Min	Typ	Max	
t_W	MCLK pulse width			30			ns
T_{SD1}	Data setup time	$\overline{IV0}$-$\overline{IV7}$	↓MCLK	35			ns
T_{SD2}	$\overline{ME}$ setup time	$\overline{ME}$	↓MCLK	30			ns
T_{SD3}	SC setup time	SC	↓MCLK	30			ns
T_{SD4}	WC setup time	WC	↓MCLK	30			ns
T_{HD1}	Data hold time	↓MCLK	$\overline{IV0}$-$\overline{IV7}$	0			ns
T_{HD2}	$\overline{ME}$ hold time	↓MCLK	$\overline{ME}$	0			ns
T_{HD3}	SC hold time	↓MCLK	SC	0			ns
T_{HD4}	WC hold time	↓MCLK	WC	0			ns
T_{PD3} (Note)	$\overline{IV}$ propagation delay	$\overline{IV}$	$D0_A$-$D7_A$/$D0_B$-$D7_B$			45	ns
T_{OE}	Output enable	$\overline{ME}$, SC, or WC	$\overline{IV0}$-$\overline{IV7}$			30	ns
T_{OD}	Output disable	$\overline{ME}$, SC, or WC	$\overline{IV0}$-$\overline{IV7}$			20	ns

Note:
Measured with MCLK = High and control signals of the primary port set for output data from the same register.

FLOPPY DISK FORMATTER/CONTROLLER

Originally published by Signetics January 1984

FEATURES

- **Single or double density encoding/decoding**
- **On-chip data separator**
- **Programmable:**
  - **FM, MFM, and M²FM encoding/decoding**
  - **Preamble Polarity**
  - **Data transfer rate**
  - **Address mark encoding/decoding**
  - **Sector length**
  - **Output port (7-bits disk command)**
  - **Input port (5-bits disk status)**
- **Write Precompensation with on/off control**
- **On-chip phase lock loop**
- **CRC generator with software-controlled error correction capabilities**
- **40-pin package**
- **+5 volt operation**

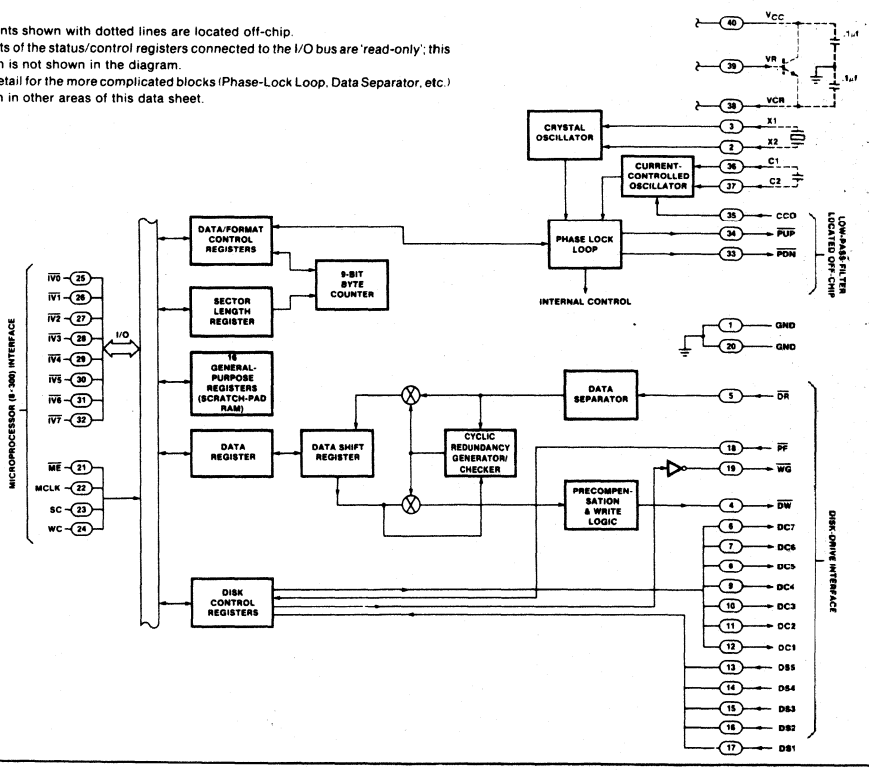
PRODUCT DESCRIPTION

The Signetics 8X330 Floppy Disk Formatter/Controller is a monolithic peripheral device of the 8X300 Family. The chip uses Bipolar-Schottky/I²L-Technology and some very unique features to provide 8X330 customers with a competitive edge in both simple and complicated disk-controller designs. The competitive advantage is measurable in terms of "systems parts count", "error correction capabilities", and "overall design concepts" that are applications oriented. Except for a crystal, a capacitor, an external transistor acting as a series-pass element for the on-chip voltage regulator, an active low-pass filter, and an optional off-chip voltage controlled oscillator (refer to Features and Option), the 8X330 contains all processing circuits and the required control logic to encode/decode double-density (MFM/M²FM) and single-density (FM) codes. Even the data-separation and write-precompensation logic are located on the chip; in addition, 16-bytes of scratch-pad RAM are provided for storage of various control/status parameters.

OPTION: External Voltage Controlled Oscillator (VCO). For critical applications, window margins can be improved by as much as 6% with the use of an external VCO.

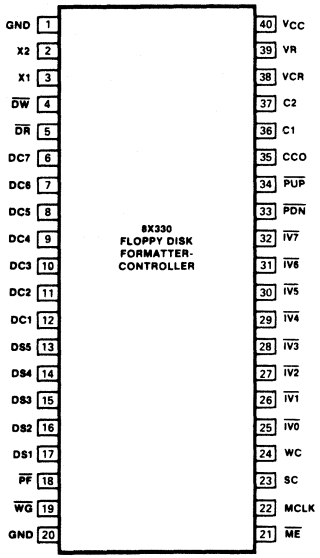
BLOCK DIAGRAM

NOTES
1. Components shown with dotted lines are located off-chip.
2. Certain bits of the status/control registers connected to the I/O bus are 'read-only'; this distinction is not shown in the diagram.
3. Greater detail for the more complicated blocks (Phase-Lock Loop, Data Separator, etc.) are shown in other areas of this data sheet.



8X330 PACKAGE/PIN DESIGNATIONS

N PACKAGE



TOP VIEW

ORDER NUMBER

N8X330N

PIN NO.	MNEMONIC & DEFINITION		FUNCTION
1, 20	GND	Ground	Circuit ground
2, 3	X1, X2	Crystal inputs	Inputs from a crystal that determines frequency of an on-chip crystal oscillator.
4	$\overline{DW}$	Data write	A series of negative-going pulses transmitted to the disk drive. The data write signal produces pulses (with precompensation, if required) for data and clock in accordance with the applicable encoding rules (FM, MFM or M²FM).
5	$\overline{DR}$	Data read	Negative-going pulses transmitted from the disk drive to a Schmitt-trigger input of the 8X330; these pulses represent encoded data and clock from the disk media.
6-12	DC1-DC7	Disk commands	Seven outputs from the 8X330 that allow general-purpose control, of one or more disk drives.
13-17	DS1-DS5	Disk status	Five general-purpose Schmitt-trigger inputs from the disk drive (or drives) that provide status information for the 8X300.
18	$\overline{PF}$	Power fail	Schmitt-trigger input from external logic that is active (low) when the "user-sensed" power supply voltage drops below a predetermined value.
19	$\overline{WG}$	Write gate	When active (low), this 40-milliampere open-collector output enables writing to the disk media. When $\overline{PF}$ is low, the write gate is inhibited during periods of power supply uncertainty.
21	$\overline{ME}$	Master enable	When this input signal is active (low), the 8X330 can be accessed and enabled by the 8X300. (Refer to the $\overline{LB}$ and $\overline{RB}$ pinout discriptions of the 8X300 for further detail.)
22	MCLK	Master clock	When active high and with $\overline{ME}$ in the active-low state, this input signal provides a means whereby the I/O output from the 8X300 is interpreted as an enabling address (provided there is an address match) or as input data (if one of the 8X330 registers has already been selected).
23	SC	Select command	When this signal is active (high), the information output on pins $\overline{IV0}$-$\overline{IV7}$ of the 8X300 is interpreted as an address input by the 8X330.
24	WC	Write command	When this signal is active (high), the information output on pins $\overline{IV0}$-$\overline{IV7}$ of the 8X300 is interpreted as input data by the 8X330.
25-32	$\overline{IV0}$-$\overline{IV7}$	Input/output lines	Eight three-state input/output lines that provide bidirectional data transfers between the 8X300 and the enabled I/O device; $\overline{IV7}$ is the *Least Significant Bit.*
33	$\overline{PDN}$	Pump down output	Open-collector output of on-chip phase detector which indicates (by a negative-going, quantized, pulse-width modulated signal) that internal CCO frequency is too high.
34	$\overline{PUP}$	Pump up output	Open-collector output of on-chip phase detector which indicates (by a negative-going, quantized, pulse-width modulated signal) that internal CCO frequency is too low.
35	CCO	Frequency Control input for Current-Controlled Oscillator	Variable input current from external low-pass filter that controls the frequency of the oscillator.
36-37	C1, C2	Capacitor input terminals	Inputs for capacitor that determines center frequency of the current-controlled oscillator.
38	VCR	Regulated supply voltage	DC voltage input from emitter of external series-pass transistor; this voltage powers internal logic of chip.
39	VR	Reference voltage	Reference voltage output to base of series-pass transistor; this reference controls VCR.
40	V_{CC}	Supply voltage	+5 volt power.

SYSTEM INTERFACE

A typical floppy disk controller using an 8X300 microcontroller and the 8X330 is shown in Figure 2. The non-shaded portion of this particular configuration can service the command, status, and input/output requirements of two double-sided disk drives and, under software supervision, the system can read/write single-density (FM) or double-density (MFM/M²FM) codes. Interface requirements are simple—on one hand, consisting of the 8X300 microcontroller and, on the other, the two disk drives. All 8X330 control and data registers directly linked to the microprocessor interface (Figure 1) are addressable and appear to the 8X300 as simple I/O ports; a 13-bit address bus and a 16-bit instruction bus provide communications between the 8X300 and up to 8K of microprogram storage.

The disk-drive interface consists of seven (7) output control lines (DC1-DC7), five (5) input status lines (DS1-DS5), a write gate ($\overline{WG}$), a data- write output ($\overline{DW}$), and a data-read input ($\overline{DR}$). The twelve command/status lines are not dedicated; thus, the user can assign system functions to best suit a given application. As shown in Figure 2, all control lines except $\overline{WG}$ are buffered to accommodate a reasonable distance between the controller and the disk media; the Write Gate, being a 40-milliampere output, requires no buffering.

As shown by the shaded part of Figure 2, the control and status lines can be expanded with peripheral hardware—the 8T32 (in this example) being only one method of implementation. Using this particular technique, one I/O port is totally dedicated to output control, whereas, the other port is totally dedicated to input status. With additional hardware and supporting software, the disk-drive system can be expanded without limit; however, from a point of being practical, five or six drives is sufficient for most applications. By using the programmable features of the 8X330, the user can emphasize and prioritize those system parameters that are most important—economics, reliability and/or speed.

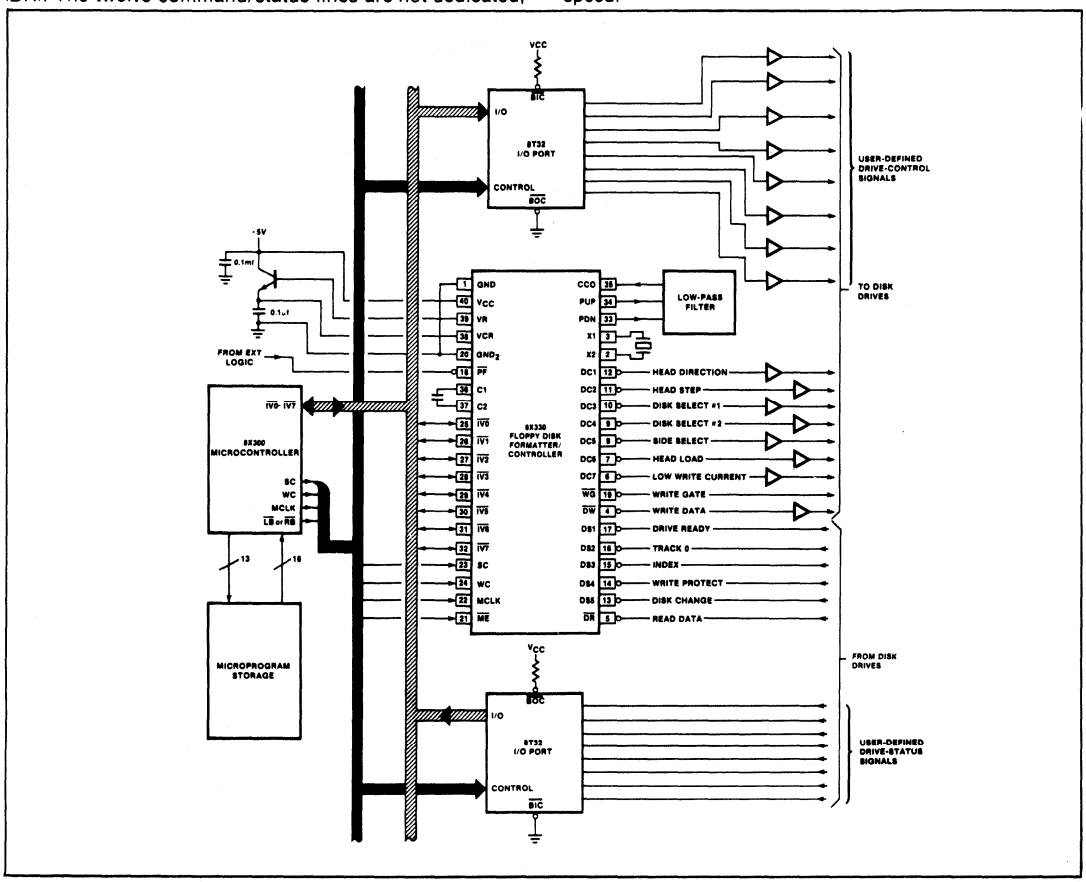


Figure 2. Typical Interface Using an 8X300 Microcontroller

FUNCTIONAL OPERATION

As shown in Figure 2, the interface between the 8X300 and 8X330 consists of twelve (12) lines—$\overline{IV0}$-$\overline{IV7}$, SC, WC, MCLK, and $\overline{ME}$; the Master Enable ($\overline{ME}$) input (pin 21) is driven from either the $\overline{LB}$ (Left Bank) or $\overline{RB}$ (Right Bank) output of the 8X300. An expanded view of this interface is shown in Figure 3 and, as indicated, the 8X330 appears as a number of addressable registers (110_8-127_8 and 132_8-137_8) under input/output control of the 8X300. These registers are used for general-purpose storage, data-transfer operations, disk commands, disk status, and various control functions. Design-oriented information for these registers and other data-processing/logic functions of the 8X330 are described in the paragraphs that follow; in all of these registers, bit 0 is the Most Significant Bit (MSB).

NOTE
When power is first applied to the 8X330, the Disk Command lines (DC1-DC7), the Write Gate ($\overline{WG}$) output, and contents of Command/Status Register #2 (CSR #2) are set to 1 (high). The wakeup state of all other bits is undefined.

General-Purpose Register File

This general-purpose (scratch pad) memory is directly accessible by the 8X300 and is used to store system variables such as track address, sector address and other necessary parameters. The sixteen 8-bit registers (110_8-127_8) provide sufficient on-chip memory to accommodate a minimum of two disk drives; the maximum number of drives that this non-dedicated memory file can support depends on several factors—system configuration, reliability requirements, economic constraints, and so on. Because of the on-chip file, all other system memory can be dedicated to the purpose of handling data to-and-from the disk media.

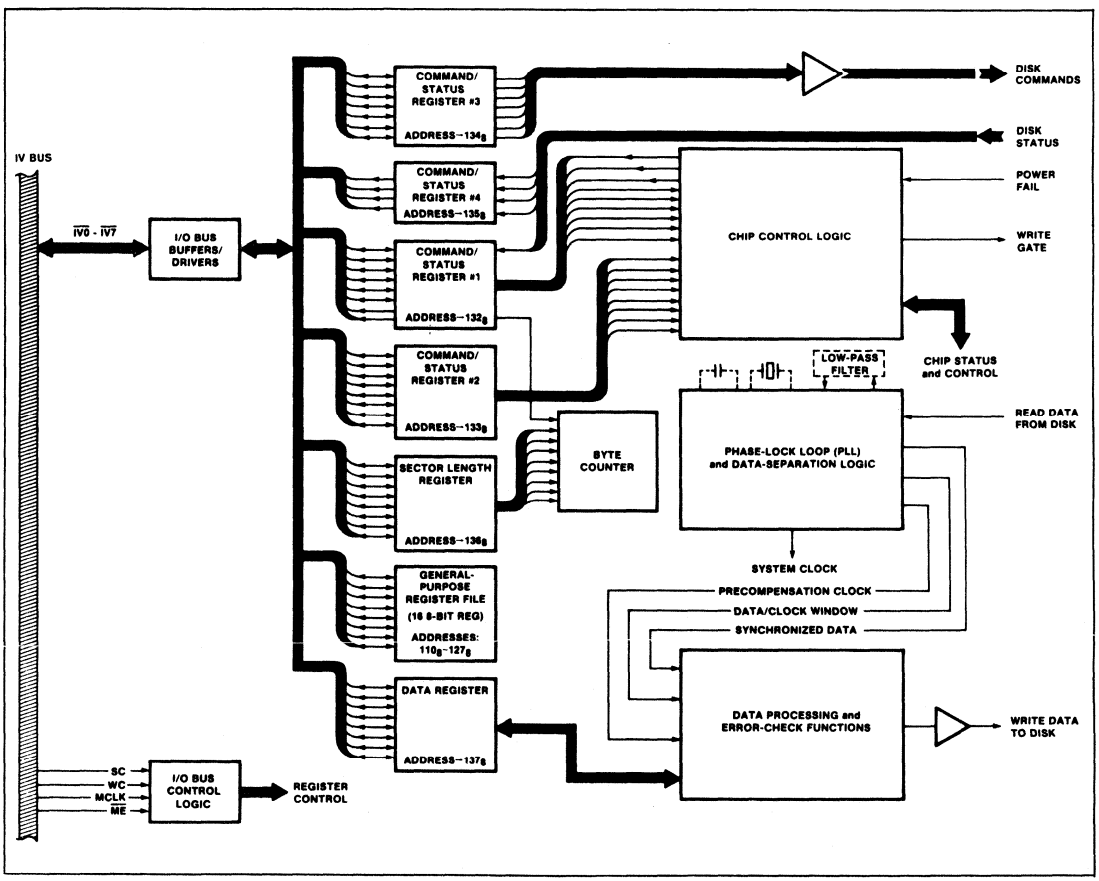


Figure 3. An Expanded View of the 8X330/8X300 Interface

Command/Status Register #1 (CSR 1/Address 132_8)

The disk status (read) or disk-command (write) contents of this register are interpreted as follows; unless otherwise indicated, all bits of CSR 1 are read/write from the I/O bus.

Bit 0 (Write Gate Enable)
Enables write gate output ($\overline{WG}$/pin 19) to disk drive(s)—the write gate ($\overline{WG}$) cannot be enabled unless the PF input pin (18) is high. When the WGE bit is set to 0, the $\overline{WG}$ ouput pin is low (enabled); when WGE is set to 1, the $\overline{WG}$ output is high (disabled). If the PF input goes low while the $\overline{WG}$ output is low, the $\overline{WG}$ output will go high and the Write Gate Enable bit is reset to 1.

Bit 1 (CRC Enable)
When set to 1, permits internal CRC register to compute remainders on the data stream in either read or write modes of operation. When set to 0, the CRC register becomes the source of data. A change in the CRC Enable bit does not become effective until the "next BYTRA flag appears" following the command bit change—refer to description of CSR 1/Bit 6.

Bit 2 (Data Register Control)
When set to 0, contents of data register consists of interleaved data-and-clock bits; starting from the MSB ($\overline{IV0}$) position, register contents are: Clock 1, Data 1, Clock 2, Data 2, Clock 3, Data 3, Clock 4, and Data 4. When writing an address mark, the appropriate data/clock pattern is loaded into the data register by the 8X300. Since each byte of data from the processor becomes an interleaved pattern (4-bits of data and 4-bits of clock) in the 8X330 data register, two bytes from the processor are required to write each full byte of address mark to the drive—eight bit cells with each cell containing a possible data and/or clock transition, or a total of 16 bit positions. When writing address marks, the normal on-chip clock insertion circuitry of the 8X330 is inhibited; thus, the user is free to define any clock/data pattern for the address mark.

When reading address marks, the data register is loaded with data and clock representing four bit cells from the disk media. The information in the data register can then be compared with the expected address mark by the 8X300 on a nibble-by-nibble basis. When the DRC bit is set to 1, the data register contains separated data (no clocks). A state change in this bit does not become effective until the "next BYTRA flag appears" following the state-change command.

Bit 3 (Sync Enable)
The Sync Enable bit allows the on-chip data separator to obtain bit and byte synchronization; this bit also controls initialization of the CRC Register. With the 8X330 in Read mode and with Bit 3 set to 0, bit synchronization occurs. The Preamble field is assumed to be all "zeroes" or all "ones" as determined by the Preamble Select bit (CSR 2/Bit 4).

When the proper number of preamble bytes, as determined by the disk-control program, have been found, the Sync Enable bit should be changed, under program direction, to a 1. This puts the 8X330 in the Address-Mark search mode. Accordingly, all bits of the CRC Register are preset to 1, the BYte TRAnsfer flag is inhibited, and the 8X330 examines the data stream for an Address Mark. The Address Mark is detected by observing the data and clock bits to find a change in the normal Preamble pattern. Byte synchronization is achieved by assuming that the change occurred in the bit cell determined by two Bit Select bits (CSR 2/Bits 2 and 3).

When the pattern change is found indicating the start of an Address Mark, the 8X330 starts CRC computation and synchronizes BYTRA to the byte boundaries. Note that the 8X330 presumes an Address Mark by finding a change in the preamble pattern; however, it is up to the 8X300 to read the Address Mark and to establish its validity or non-validity.

In write mode, setting the Sync Enable bit to 0 presets all bits of the CRC Register to 1. Setting the Sync Enable bit to 1 allows CRC computation to begin at the next byte boundary.

Bit 4 (Load Counter)
When set to 1, transfers 8-bits of data from Sector Length register and 1-bit (MSB) of data from Byte Counter (refer to next description) to 9-bit Byte Counter. Loading of the 9-bit Byte Counter is effective one bit-cell time after the Load Counter bit is set to 1. In both the read and write modes of operation, the Byte Counter is incremented by BYTRA. The Load Counter bit is self-clearing and always returns a 0 when read.

NOTE
The Load Counter bit must be set one or more instruction cycles *after* setting the Byte Counter MSB, that is, bits 4 and 5 of CSR 1 cannot be set during th same instruction cycle.

Bit 5 (Byte Counter MSB)
This bit is used to set and monitor the state of the ninth (MSB) bit in the Byte Counter; reading this bit always returns the current state of MSB in the Byte Counter. The MSB of the Byte Counter is set to the value of CSR 1/Bit 5 when the Load Counter bit (CSR 1/Bit 4) is asserted—refer to preceding description.

NOTE
The Byte Counter MSB must be set one or more instruction cycles *before* the Load Counter bits—bits 4 and 5 of CSR 1 cannot be set during the same instruction cycle.

Bit 6 (BYTRA)
During a disk read operation, the BYte TRAnsfer flag is automatically set to 0 when 8-bits of information are transferred from the Data Shift Register to the Data Register —see Figure 1. During a disk write operation, BYTRA is automatically set to 0 when 8-bits are transferred from the Data Register to the Data Shift Register. BYTRA (a read-

only bit) is reset to a 1 when the Data Register (address 137_8) is selected by the user's program. During read/write operations, the 1-to-0 transition of the BYTRA flag increments the Byte Counter to keep count of bytes read or bytes written. All read-only bits of the 8X330 are designed to remain stable during the monitor period; thus, to read a status change of BYTRA, Disk-Status bit, the Byte Counter MSB, or other read-only bit requires a two-instruction loop similar to:

```
TEST    SEL   CSR 1
        NZT   BYTRA,   TEST
```

Bit 7 (Disk Status 1)
Reflects state (0 or 1) of input DS1 (pin 17); this is a user-definable read-only bit.

NOTE
A high input on any one of the Disk Status lines of the 8X330 is read by the 8X300 program as a logical 1 and a low input on the status lines is read as a logical 0.

Command/Status Register #2 (CSR 2/Address 133_8)

The disk status (read) or disk-command (write) contents of this register are interpreted as follows:

Bit 0 (Precompensation Enable)
This command bit determines whether or not precompensation is applied to the data stream being written onto the disk. When set to 0, precompensation is inhibited. When set to 1 and with double-density encoding, write precompensation is applied to the following data/clock bit patterns:

Precomp Time	Data/Clock Pattern (in Data Shift Reg)			Bit Being Written	Bits Already Written to Disk		
2T (Late)	0	1	**0**	1	0	0	0
2T (Late)	0	1	**0**	1	0	0	1
2T (Early)	1	0	**0**	1	0	1	0
2T (Early)	0	0	**0**	1	0	1	0

where, $T = \frac{1}{\text{crystal frequency}}$ if bit 7 of CSR 2 ($1/2F$) = 1

$T = \frac{2}{\text{crystal frequency}}$ if bit 7 of CSR 2 ($1/2F$) = 0

Bit 1 (Read Mode)
When set to 0, the 8X330 reads data from the disk and transfers it to the Data Register; when set to 1, data from the Data Register is transferred to the disk, provided the Write Gate Enable bit (CSR1/Bit 0) is set to 1. With WGE set to 0 and the Read Mode bit set to 1, the current-controlled oscillator is forced to lock onto the crystal oscillator; this technique is used during a data-read operation to ensure rapid acquisition of the disk data.

Bits 2,3 (Bit Selects 1 and 0)
Together with the Sync Enable (CSR 1/Bit 3), these two bits allow the user to establish byte boundaries for the data stream; this is done in the following way. After bit synchronization is established, and the preamble pattern is verified, the 8X330 looks for a change in the normal preamble pattern. As shown in the following truth table, Bit Select 1 (Bit 2) and Bit Select 0 (Bit 3) identifies the bit cell within the first nibble of the first Address-Mark byte in which the first deviation from the normal preamble is expected. BYTRA is always referenced to bit cell 0.

BS 0	BS 1	Bit Cell
0	0	0
0	1	1
1	0	2
1	1	3

Bit 4 (Preamble Select)
This bit is used only for bit sychronization—refer to CSR 1/Bit 3. With Bit 1 of CSR 2 set to 0 (Read Mode) and the Preamble Select bit set to 0, the preamble field is assumed to be all zeroes; with the Preamble Select it set to 1, the preamble field is assumed to be all ones. In either case, preamble validity is determined by the 8X300.

Bits 5,6 (E1 and E2)
Together, E1 and E2 select the encoding scheme used to write data on the disk—refer to truth table that follows.

E1	E2	Encoding Scheme
1	X	FM
0	0	MFM
0	1	M²FM

x = don't care

Bit 7 (1/2F)
This bit allows the data transfer rate to be changed without modification of the frequency-selective components in the data-separation logic; thus, differences in data transfer rates between standard-and-mini floppies can be accommodated via software—no component or other hardware changes. Assuming an 8 MHz crystal and with the 1/2F bit set to 1, the data transfer rate is 250K-bits per second in the single-density (FM) mode and 500K-bits per second in the double-density (MFM/M²FM) mode. When set to 0, the transfer rates are halved—125K-bits and 250K-bits, respectively. When using frequencies other than 8 MHz, the data transfer rate is determined as follows:

Bit 7 ($1/2F$)	Single-Density (FM)	Double-Density (MFM/M²FM)
0	$\frac{\text{xtal freq}}{64}$	$\frac{\text{xtal freq}}{32}$
1	$\frac{\text{xtal freq}}{32}$	$\frac{\text{xtal freq}}{16}$

Command/Status Register #3 (CSR 3/Address 134_8)

This register contains seven bits (Bit 0 through Bit 6) which determines the state of the disk-command outputs; writing to Bit 7 has no effect and reading Bit 7 always returns a zero. When a logical "1" is specified by the 8X300 program for a given disk-command line, a high will appear at the output of the 8X330 for that particular command line. Each bit and the output pin it controls are summarized below.

Bit (CSR 3)	Control Function	Pkg Pin No.
0	DC1 Output	12
1	DC2 Output	11
2	DC3 Output	10
3	DC4 Output	9
4	DC5 Output	8
5	DC6 Output	7
6	DC7 Output	6

Command/Status Register #4 (CSR 4/Address 135_8)

This register contains four bits (Bit 0 through Bit 3) which reflect the state of the disk-status inputs to the 8X330; reading all other bits (4 through 7) always returns a zero. These read-only bits and the reflected status they represent are as follows; the information specified by notation for Bit 7/CSR 1 is applicable to these input lines.

Bit (CSR 4)	Control Function	Pkg Pin No.
0	DS2 Input	16
1	DS3 Input	15
2	DS4 Input	14
3	DS5 Input	13

Sector Length Register—Address 136_8

This register contains the load value for the lower eight (LSBs) bits of the Byte Counter. Data is transferred from the Sector Length Register to the Byte Counter under control of Load Counter Bit in CSR 1. When the contents of this register are transferred to another location via a read or write commands, the original holding of data is not lost; thus, if the same data is to be used more than once, a repetitive read or write can be implemented without reloading the register.

Data Register—Address 137_8

Together with the Data Shift Register, the Data Register is used for bidirectional transfer of data between the 8X330 and the I/O bus. All transfers to-and-from this register are made in conjunction with Bit 6 (BYTRA—Byte Transfer Flag) of CSR 1. When the Data Register Control bit (CSR 1/Bit 2) is set to 0, the content of this register is interleaved with four bits of data and four bits of clock. When data is transferred from the Data Register to the Data Shift Register, the original content of the Data Register is not lost.

Phase Lock Loop (PLL) and Data Separation Logic

An expanded view of the phase-lock loop and the data-separation logic is shown in Figure 4. Basically, the PLL consists of two counters, a phase detector, and a feedback loop containing a low-pass filter (off-chip) that controls a phase-locked oscillator (CCO). In simplified form, the data-separation logic consists of data flip-flops (pulse synchronizer) and other circuits required to separate data and clock transitions. In the read mode, the output of the phase-locked oscillator (CCO) is applied to the clock inputs of counter #1, counter #2, and the pulse synchronization circuits. Essentially, the frequencies of the two counters are identical (phase relationships may or may not be identical); to maintain proper frequencies and to continuously correct for any phase deviations, the following actions occur.

Preset values which represent, respectively, nominal midpoints of the clock and data windows are present at counter #2 and, when an output appears at the pulse synchronizer, these preset values are entered. The count sequence for both counters is from "0 to F"; hence, the phase difference between Carry 1 (counter #1) and Carry 2 (counter #2) actually corresponds to any phase deviation between the CCO and the synchronized data from the disk. The phase detector measures the phase difference between the two carry inputs and produces a series of quantized pulses whose widths are proportional to the phase error at the end of each counting cycle. After integration by the low-pass filter, a current proportional to the phase error is applied to the current-controlled oscillator. Accordingly, the CCO is driven in a direction (pump-up or pump-down) to correct any phase difference between the synchronized disk data and the feedback-controlled clock. Phase detector characteristics for both single-and-double density formats are shown in Figures 5 and 6.

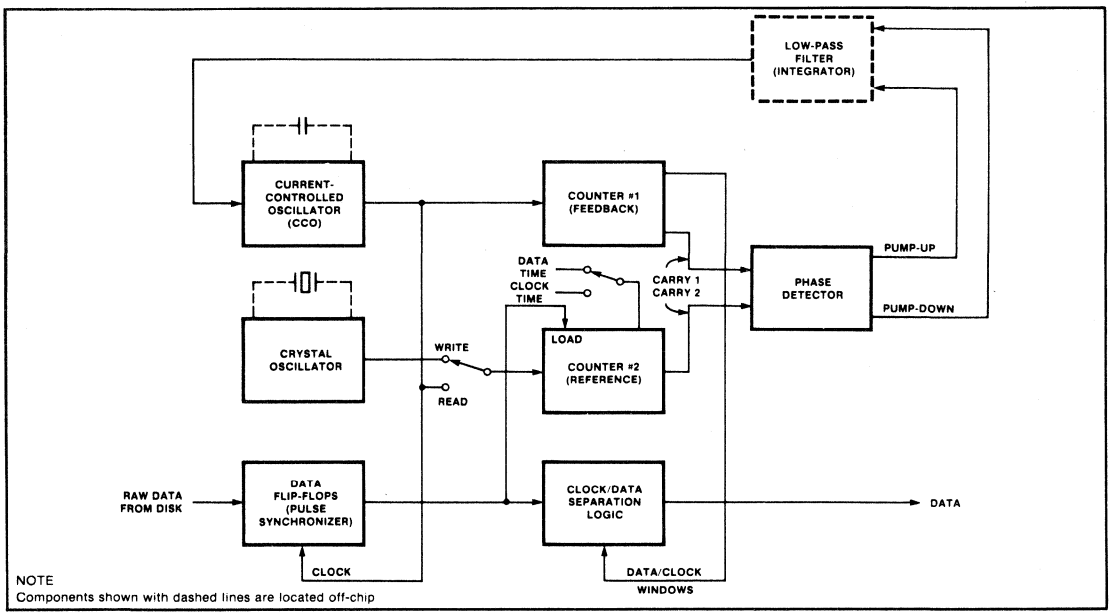


Figure 4. Simplified Block of Phase-Lock Loop

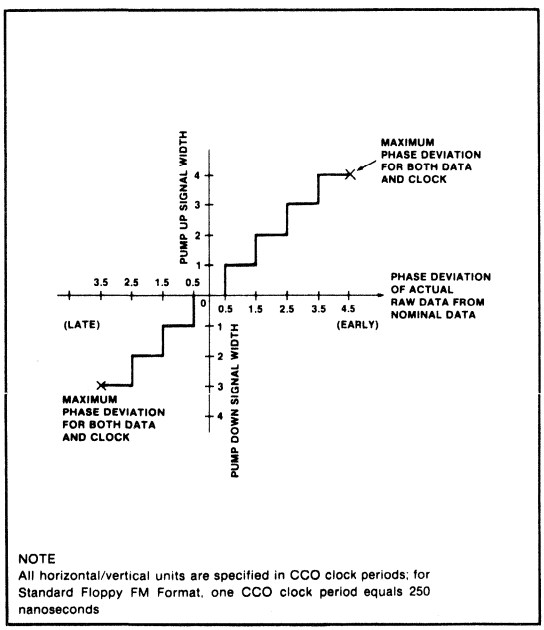


Figure 5. Phase Detector Characteristic for Single-Density (FM) Format

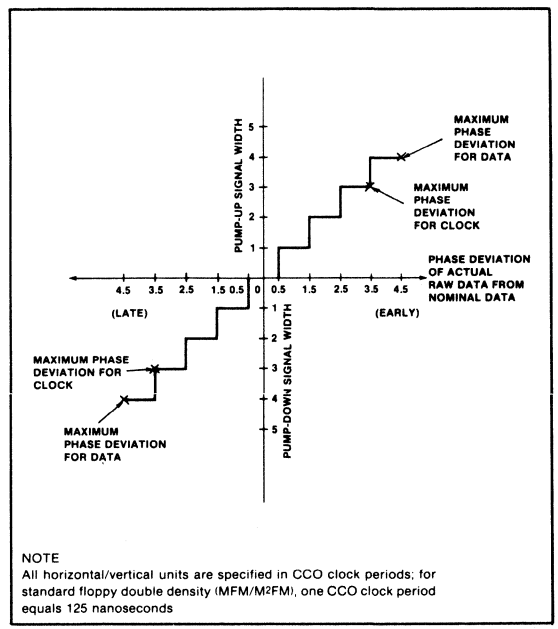


Figure 6. Phase Detector Characteristic for Double-Density (MFM/M²FM) Format

Data Processing and Error-Check Functions

These functions of the 8X330 are summarized in Figures 7 and 8. The read/write operations are software-controlled by previously-described bits of command/status registers CSR1 and CSR2. For the sake of simplicity, control lines and much of the control logic associated with the data processing and error-check functions are omitted in the read/write diagrams.

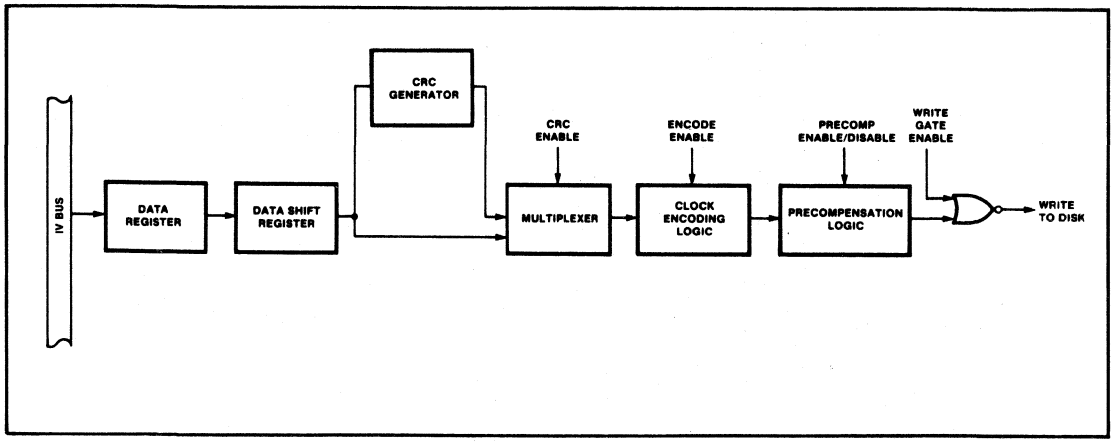


Figure 7. Simplified Block of Data Processing and Error Check Functions—Write Mode

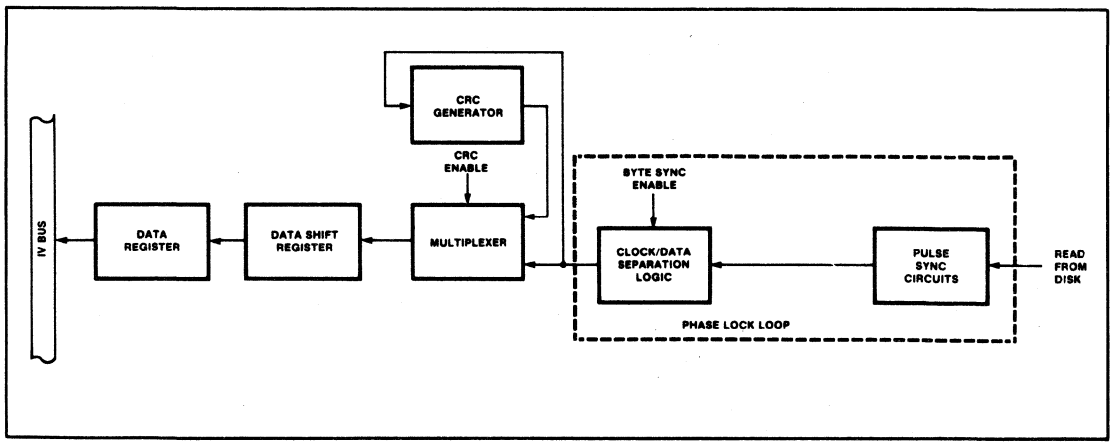


Figure 8. Simplified Block of Data Processing and Error Check Functions—Read Mode

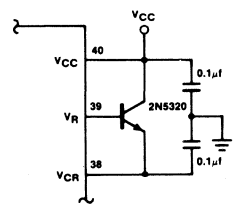
DC CHARACTERISTICS V_{CC} = 5V (±5%), T_A = 0°C to 70°C

PARAMETER		TEST CONDITIONS	LIMITS			UNITS	COMMENTS
			Min	Typ	Max		
V_{IH}	High level input voltage		2.0		V_{CC}	V	For all inputs except X1, X2, C1, C2, CCO, and V_{CR}
V_{IL}	Low level input voltage		−1.0		0.8	V	
V_{CC}	Supply voltage		4.75	5.0	5.25	V	5V (±5%)
V_{CR}	Regulator voltage	V_{CC} = 5V		3.1		V	From series-pass transistor
V_{CL}	Input clamp voltage	V_{CC} = Min I_{IN} = −5mA	−1.0			V	Inputs X1, X2, C1, C2, and CC0 do not have internal clamp diodes.
V_{OH}	High level output voltage	V_{CC} = Min; I_{OH} = −0.4mA	2.7			V	DC1 through DC7 (Pins 6-12) & $\overline{DW}$ (Pin 4)
		V_{CC} = Min; I_{OH} = −3mA	2.4			V	$\overline{IV0}$-$\overline{IV7}$ (Pins 25-32)
V_{OL}	Low level output voltage	V_{CC} = Min; I_{OL} = 8mA			0.5	V	DC1 through DC7 (Pins 6-12); $\overline{PUP}$, $\overline{PDN}$ (Pins 33, 34); $\overline{DW}$ (Pin 4)
		V_{CC} = Min; I_{OL} = 16mA			0.55	V	$\overline{IV0}$-$\overline{IV7}$ (Pins 25-32)
		V_{CC} = Min; I_{OL} = 40mA			0.55	V	$\overline{WG}$ (Pin 19)
I_{CEX}	Open-collector leakage current with output set to 1.	V_{CC} = Min; V_{OUT} = V_{CC}			100	µA	$\overline{WG}$ (Pin 19); $\overline{PUP}$ (Pin 34); $\overline{PDN}$ (Pin 33)
I_{IH}	High level input current	V_{CC} = Max; V_{IN} = 2.7V			20	µA	DS1-DS5 (Pins 13-17); $\overline{PF}$ (Pin 18); $\overline{DR}$ (Pin 5)
					40	µA	$\overline{ME}$ (Pin 21); MCLK (Pin 22); SC (Pin 23); WC (Pin 24)
		V_{CC} = Max; V_{IN} = 5.25V; CCO (Pin 35) input current = 0mA			4	mA	With C1 (Pin 36) under test, C2 (Pin 37) is open and, vice-versa.
		V_{CC} = Max; V_{IN} = 5.25V CCO (Pin 35) input current = 1mA			2	mA	
		V_{CC} = Max; V_{IN} = 0.6V			4	mA	With X2 (Pin 2) under test, X1 (Pin 3) is open and, vice-versa.
		V_{CC} = Max; V_{IN} = 4.5V			50	µA	$\overline{IV0}$ - $\overline{IV7}$ (pins 25-32)
V_{CCO}	Input voltage for current-controlled oscillator	V_{CC} = 5V; T_A = 25°C CCO input current (Pin 35) = 300 µA		750		mV	

DC CHARACTERISTICS (Cont'd) V_{CC} = 5V (±5%), T_A = 0°C to 70°C

PARAMETER		TEST CONDITIONS	LIMITS			UNITS	COMMENTS
			Min	Typ	Max		
I_{IL}	Low level input current	V_{CC} = Max; V_{IN} = 0.4V			−400	µA	DS1-DS5 (Pins 13-17); $\overline{PF}$ (Pin 18); $\overline{DR}$ (Pin 5)
					−800	µA	$\overline{ME}$ (Pin 21); MCLK (pin 22); SC (Pin 23); WC (Pin 24)
					−4	mA	X1 (Pin 2), X2 (Pin 3), with X1 under test, X2 is open and, vice-versa.
		V_{CC} = Max V_{IN} = 0.5V			−550	µA	$\overline{IV0}$-$\overline{IV7}$ (Pins 25-32)
I_{OS}	Output short-circuit current	V_{CC} = Max; Output = "1"; V_{OUT} = "0"	−15		−100	mA	DC1-DC7 (Pins 6-12) & $\overline{DW}$ (Pin 4)
		(NOTE At any time, no more than one output should be connected to ground)	−30		−140	mA	$\overline{IV0}$-$\overline{IV7}$ (Pins 25-32)
I_{CC}	(Pin 40)	V_{CC} = Max			200	mA	
I_{CR}		V_{CC} = Max			250	mA	
I_{REG}	(Pin 39)	V_{CC} = 5V; V_{CR} = 0V & V_R = 2V	−16		−27	mA	

NOTES
1. Operating temperature ranges are guaranteed after thermal equilibrium has been reached.
2. All voltages measured with respect to ground terminal.
3. Unless otherwise specified, each test requires that V_{CR} be supplied through a series-pass transistor as shown in the accompanying drawing.



AC CHARACTERISTICS V_{CC} = 5V (±5%), T_A = 0°C to 70°C

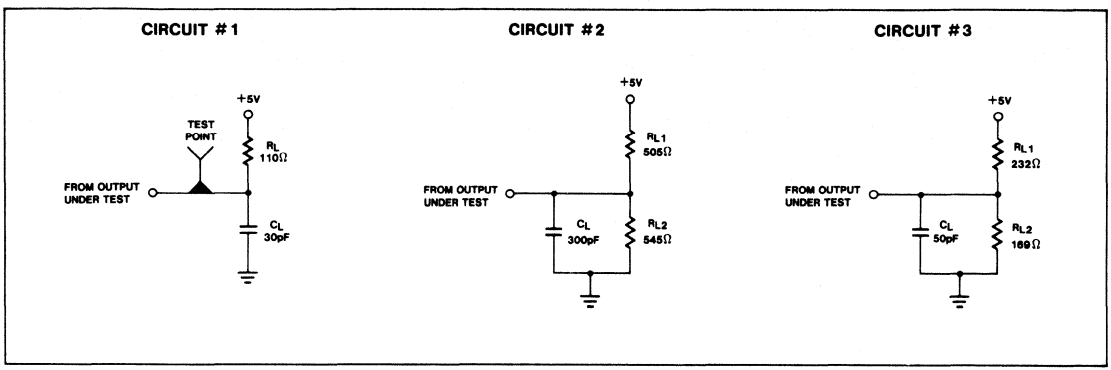
MNEMONIC	REFERENCE	INPUT	OUTPUT	Min	Typ	Max	COMMENTS
t_{PD}		$\overline{PF}$ ↓	$\overline{WG}$ ↓			60ns	Refer to Note 3 and Test Loading Circuit #1.
t_{PD}		MCLK ↑	$\overline{WG}$ ↓			100ns	
t_{PD}		MCLK ↑	$\overline{WG}$ ↑			100ns	
t_{PD}		MCLK ↑	DC1-7			70ns	Refer to Note 3 and Test Loading Circuit #2.
t_{RD}		MCLK ↑	DC1-7			70ns	
t_{pw}		$\overline{PF}$		50ns			
t_{pw}		$\overline{DR}$		50ns			
t_{pw}			$\overline{DW}$				Note 1
t_{SETUP}	SC	Input on DS1-5		55ns			Note 2
t_{SETUP}	WC	Input on DS1-5		55ns			Note 2
t_{SETUP}	$\overline{ME}$	Input on DS1-5		55ns			Note 2
t_{HOLD}	SC	Input on DS1-5		0ns			Note 2
t_{HOLD}	WC	Input on DS1-5		0ns			Note 2
t_{HOLD}	$\overline{ME}$	Input on DS1-5		0ns			Note 2
t_{OE}—$\overline{ME}$, SC & WC		$\overline{ME}$, SC, WC	I/O bus			25ns	Refer to Test Loading Circuit #3.
t_{OD}—$\overline{ME}$, SC & WC		$\overline{ME}$, SC, WC	I/O bus (three-state)			30ns	
t_w (MCLK pulse width)		MCLK		45ns			
t_{SD} (data setup time)	MCLK	I/O bus		50ns			
t_{SD} ($\overline{ME}$ setup time)	MCLK	$\overline{ME}$		45ns			
t_{SD} (SC setup time)	MCLK	SC		45ns			
t_{SD} (WC setup time)	MCLK	WC		45ns			
t_{HD} (data hold time)	MCLK	I/O bus		0ns			

AC CHARACTERISTICS (Cont'd) V_{CC} = 5V (±5%), T_A = 0°C to 70°C

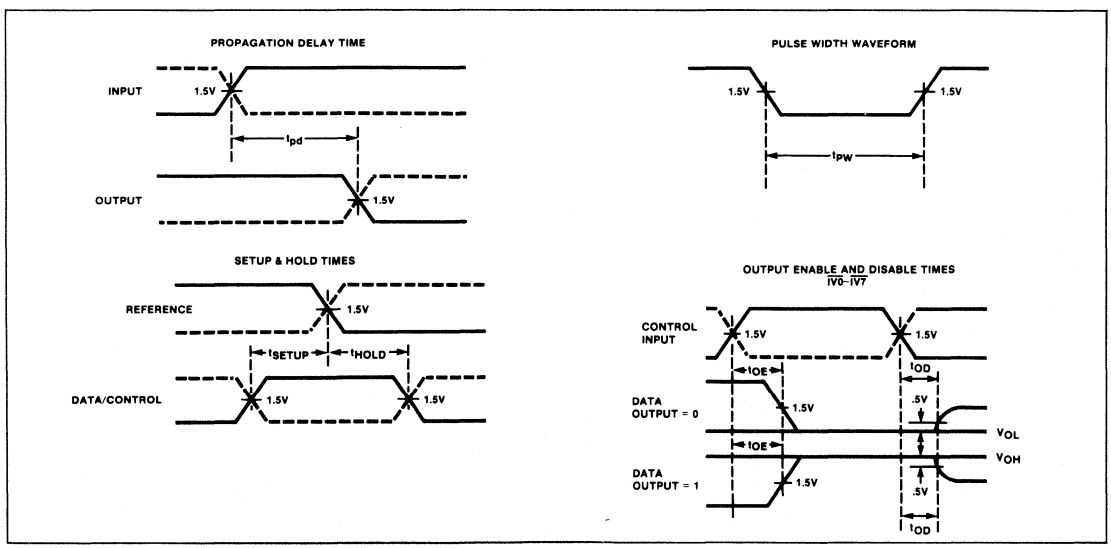
MNEMONIC	REFERENCE	INPUT	OUTPUT	Min	Typ	Max	COMMENTS
t_{HD} ($\overline{ME}$ hold time)	MCLK	$\overline{ME}$		0ns			
t_{HD} (SC hold time)	MCLK	SC		0ns			
t_{HD} (WC hold time)	MCLK	WC		0ns			

NOTES
1. Write pulse width = $2/F_{XTAL}$, that is, for 8MHz crystal, t_{pw} = 250nsec (typical)
2. Changes on DS1-5 are not stored in read mode ($\overline{ME}$ = 0, SC = 0, and WC = 0)
3. During the period when MCLK is high, measurement is made with $\overline{ME}$ = Low, SC = Low, and WC = High.

TEST LOADING CIRCUIT



TIMING DIAGRAM



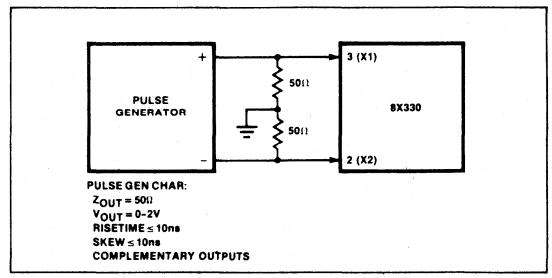
CLOCK REQUIREMENTS

Crystal Oscillator. The on-chip crystal oscillator circuit is designed for operation using an external series-resonant quartz crystal; alternately the crystal oscillator can be driven with complementary outputs of a pulse generator or interfaced to a master clock source via TTL logic—see accompanying circuits. When a crystal is used, the on-chip oscillator operates at the resonant frequency (f.) of the crystal; the crystal connects to the 8X330 via pins 3 (X1) and 2 (X2). The lead lengths of the crystal should be approximately equal and as short as possible; also, avoid close proximity to all potential noise sources. The crystal should be hermetically sealed (HC type can) and have the following electrical characteristics:

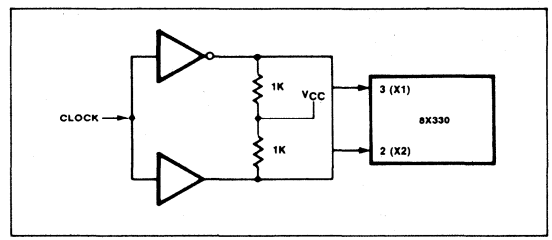
Type: Fundamental mode, series resonant
Impedance at Fundamental: 35-ohms maximum
Impedance at Harmonics and Spurs: 50-ohms minimum

When the crystal oscillator is externally-driven, typical waveforms are as follows:

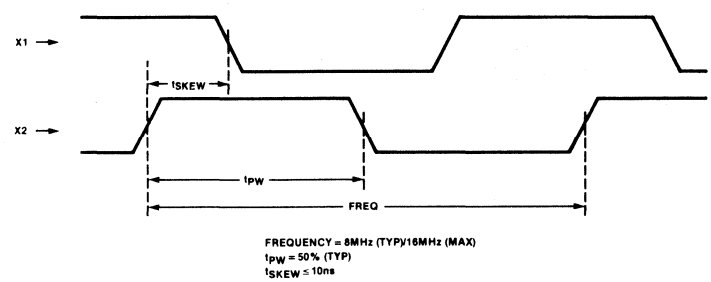
CLOCKING XTAL OSC WITH PULSE GEN



CLOCKING XTAL OSC WITH OPEN-COLLECTOR TTL

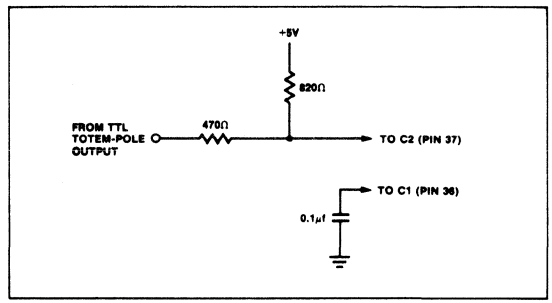


TYPICAL WAVE FORM

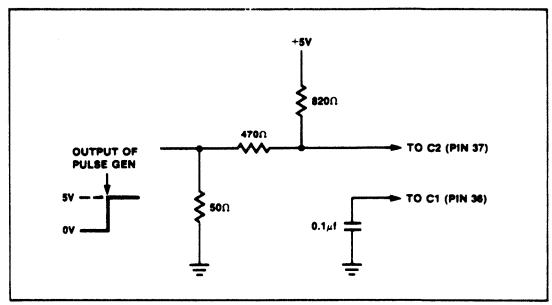


Current-Controlled Oscillator (CCO).
A non-polarized ceramic or mica capacitor is recommended for the current-controlled oscillator. The capacitor connects to the 8X330 via pins 37 (C2) and 36 (C1); lead lengths of the capacitor should be approximately the same and as short as possible. When the input current to the CCO is near zero (maximum frequency), the capacitor value should be chosen so that the high-limit rest frequency of the oscillator does not exceed 24 MHz. If the rest frequency is higher than 24 MHz, synchronization of the CCO with the crystal oscillator just prior to the read operation, may be impeded. The curves in Figure 9 (current-versus-frequency) and Figure 8 (capacitance-versus-frequency) show how these design parameters affect operation of the CCO over a temperature range of 0° C to 70° C. A suitable test circuit for verification/validation of the current-controlled oscillator is also shown in Figure 10. Like the crystal oscillator, the CCO can be driven with the TTL output of a pulse generator or interfaced to a master clock via TTL logic—see accompanying diagrams.

CLOCKING WITH OPEN-COLLECTOR TTL



CLOCKING WITH PULSE GENERATOR



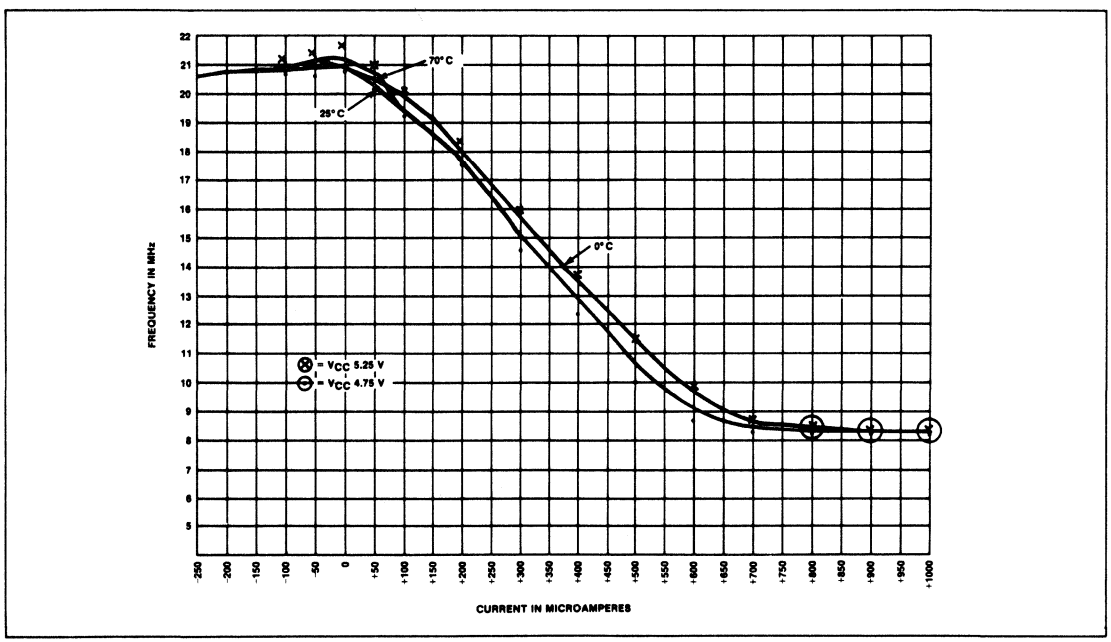


Figure 9. Current-versus-Frequency with: V_{CC} = 5V and Capacitance = 25 Picofarads

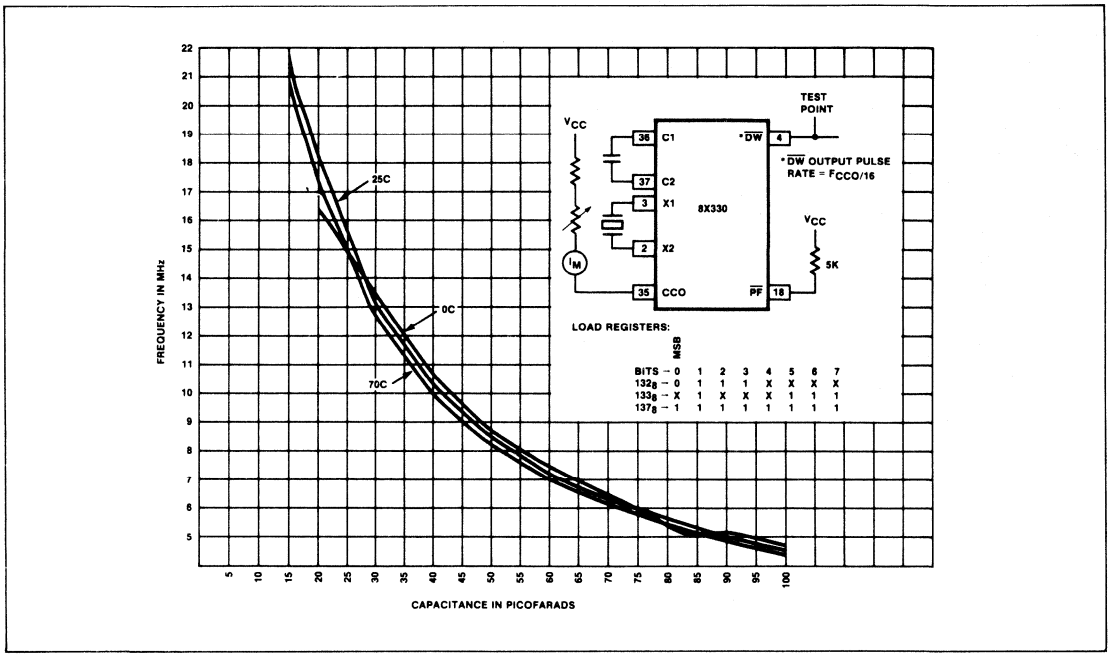


Figure 10. Capacitance-versus-Frequency with: V_{CC} = 5V, VCR = 2.5V, and I = 300μA

VOLTAGE REGULATOR

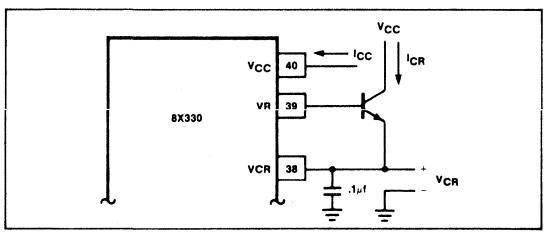
All internal logic of the 8X330 is powered by an on-chip voltage regulator that requires an external series-pass transistor. Electrical specifications for the off-chip power transistor and a typical hook-up are shown in accompanying diagrams. To minimize lead inductance, the transistor should be as close as possible to the 8X330 package and the emitter should be ac-grounded via a 0.1-microfarad capacitor.

ELECTRICAL SPECIFICATIONS

*PARAMETER	CONDITIONS	LIMITS
h_{fe}	V_{CE} = 2V	>50
V_{BEON}	V_{CE} = 5V/I_C = 500mA	<1V
V_{CESAT}	I_C = 500 mA/I_B = 50 mA	<0.5V
BV_{CEO}		>8V
f_t		>30 MHz

*Medium power NPN silicon (0°<T_A<70°C) recommended parts: 2N5320, 2N5337

TYPICAL HOOK-UP



2048-BIT BIPOLAR RAM (256 × 8)

Originally published by Signetics January 1984

FEATURES

- **On-chip address latches**
- **3-state outputs**
- **Schottky clamped TTL**
- **Internal control logic for 8X300 system**
- **Directly interfaces with the 8X300 bipolar microprocessor with no external logic**
- **May be used on left or right bank**

APPLICATIONS

- **8X300 or 8X305 working storage**

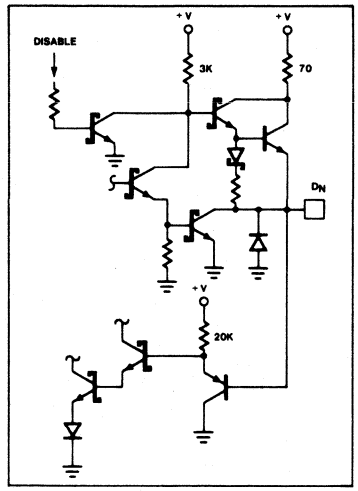
DESCRIPTION

The 8X350 bipolar RAM is designed principally as a working storage element in an 8X300 based system. Internal circuitry is provided for direct use in 8X300 applications. When used with the 8X300, the RAM address and data buses are tied together and connected to the IV bus of the system.

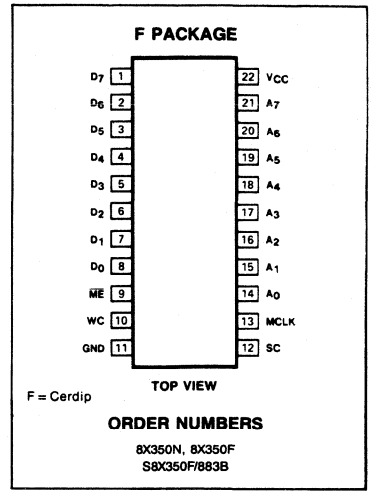
The data inputs and outputs share a common I/O bus with 3-state outputs.

The 8X350 is available in commercial and military temperature ranges. For the commercial temperature range (0°C to +75°C) specify N8X350-F, and for the miltiary temperature range (−55°C to +125°C) specify S8X350-F.

TYPICAL I/O STRUCTURE



PIN CONFIGURATION



F = Cerdip

ORDER NUMBERS

8X350N, 8X350F
S8X350F/883B

ABSOLUTE MAXIMUM RATINGS

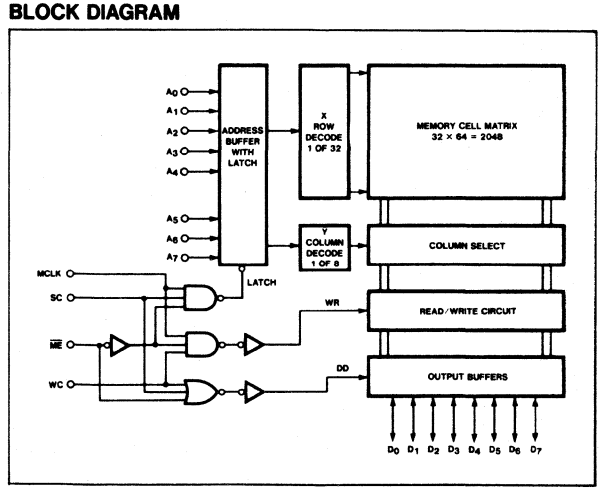
PARAMETER		RATING	UNIT
V_{CC}	Supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
	Output voltage		Vdc
V_{OH}	High	+5.5	
V_O	Off-state	+5.5	
	Temperature range		°C
T_A	Operating		
	Commercial	0 to +75	
	Military	−55 to +125	
T_{STG}	Storage	−65 to +150	

TRUTH TABLE

Note X = Don't care

MODE	$\overline{ME}$	SC	WC	MCLK	BUSSED DATA/ADDRESS LINES
Hold address Disable data out	1	X	X	X	High Z data out
Input new address	0	1	0	1	Address High Z
Hold address Disable data out	0	1	0	0	High Z data out
Hold address Write data	0	0	1	1	Data in
Hold address Disable data out	0	0	1	0	High Z data out
Hold address Read data	0	0	0	X	Data out
Undefined state[12]	0	1	1	1	—
Hold address[12] Disable data out	0	1	1	0	High Z data out

BLOCK DIAGRAM



DC ELECTRICAL CHARACTERISTICS[2]

N8X350: 0°C ≤ T_A ≤ +75°C, 4.75V ≤ V_{CC} ≤ 5.25V
S8X350: −55°C ≤ T_A ≤ +125°C, 4.75V ≤ V_{CC} ≤ 5.25V

PARAMETER		TEST CONDITIONS	N8X350			S8X350			UNIT
			Min	Typ	Max	Min	Typ	Max	
	Input voltage								V
V_{IL}	Low[1]	V_{CC} = Min			.85			.80	
V_{IH}	High[1]	V_{CC} = Max	2.0			2.0			
V_{IC}	Clamp[1,3]	V_{CC} = Min, I_{IN} = −12mA			−1.2			−1.2	
	Output voltage	V_{CC} = Min							V
V_{OL}	Low[1,4]	I_{OL} = 9.6mA			0.5			.5	
V_{OH}	High[1,5]	I_{OH} = −2mA	2.4			2.4			
	Input current								μA
I_{IL}	Low	V_{IN} = 0.45V			−100			−150	
I_{IH}	High	V_{IN} = 5.5V			25			50	
	Output current								μA
$I_{O(OFF)}$	High Z state	$\overline{ME}$ = High, V_{OUT} = 5.5 V			40			60	μA
		$\overline{ME}$ = High, V_{OUT} = 0.5 V			−100			−100	
I_{OS}	Short circuit[3,6]	SC = WC, $\overline{ME}$ = Low, V_{OUT} = 0V, Stored High	−20		−70	−15		−85	mA
I_{CC}	V_{CC} supply current[7]	V_{CC} = Max			185			200	mA
	Capacitance	$\overline{ME}$ = High, V_{CC} = 5.0V							pF
C_{IN}	Input	V_{IN} = 2.0V		5			5		
C_{OUT}	Output	V_{OUT} = 2.0V		8			8		

AC ELECTRICAL CHARACTERISTICS[2,9]

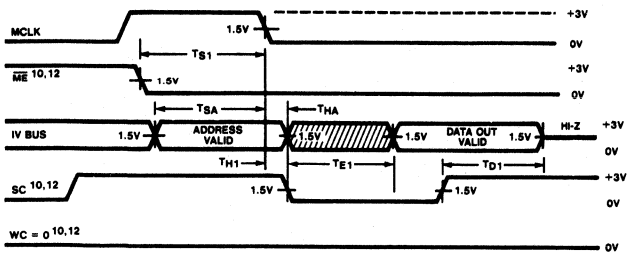
N8X350: 0°C ≤ T_A ≤ +75°C, 4.75V ≤ V_{CC} ≤ 5.25V R_1 = 470Ω, R_2 = 1kΩ, C_L = 30pF
S8X350: −55°C ≤ T_A ≤ +125°C, 4.75V ≤ V_{CC} ≤ 5.25V

PARAMETER		TO	FROM	N8X350			S8X350			UNIT
				Min	Typ	Max	Min	Typ	Max	
	Enable time									ns
T_{E1}	Output	Data out	SC−			35			40	
T_{E2}	Output	Data out	$\overline{ME}$−			35			40	
	Disable time									ns
T_{D1}	Output	Data out	SC+			35			40	
T_{D2}	Output	Data out	$\overline{ME}$+			35			40	
	Pulse width									ns
T_W	Master clock[8]			40			50			
	Setup and hold time									ns
T_{SA}	Setup time	MCLK−	Address	30			40			
T_{HA}	Hold time	Address	MCLK−	5			10			
T_{SD}	Setup time	MCLK−	Data in	35			45			
T_{HD}	Hold time	Data in	MCLK−	5			10			
T_{S3}	Setup time	MCLK−	$\overline{ME}$−	40			50			
T_{H3}	Hold time	$\overline{ME}$+	MCLK−	5			5			
T_{S1}	Setup time	MCLK−	$\overline{ME}$−	30			40			
T_{H2}	Hold time	$\overline{ME}$−	MCLK−	5			5			
T_{S2}	Setup time	$\overline{ME}$−	SC−,WC−	0			5			
T_{H1}	Hold time	SC−	MCLK−	5			5			
T_{H4}	Hold time	WC−	MCLK−	5			5			

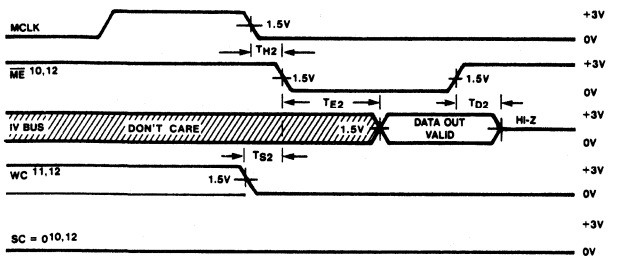
Notes on following page.

TIMING DIAGRAMS

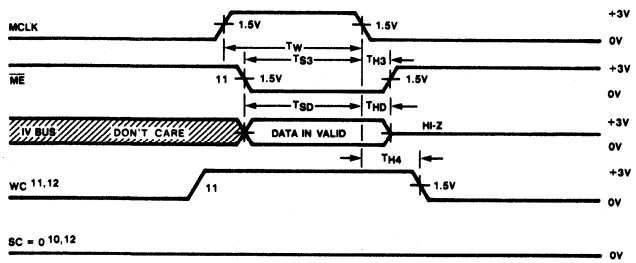
TRANSPARENT ADDRESS READ CYCLE



LATCHED ADDRESS READ CYCLE



LATCHED ADDRESS WRITE CYCLE



NOTES

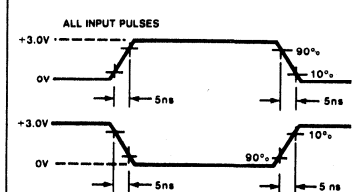
1. All voltage values are with respect to network ground terminal.
2. The operating ambient temperature ranges are guaranteed with transverse air flow exceeding 400 linear feet per minute and a 2-minute warm-up.
   Typical thermal resistance values of the package at maximum temperature are:
   θ_{JA} junction to ambient at 400fpm air flow - 50°C/watt
   θ_{JA} junction to ambient - still air - 90°C/watt
   θ_{JA} junction to case - 20°C/watt
3. Test each pin one at a time.
4. Measured with a logic low stored Output sink current is supplied through a resistor to V_{CC}.
5. Measured with a logic high stored.
6. Duration of the short circuit should not exceed 1 second.
7. I_{CC} is measured with the write enable and memory enable inputs grounded, all other inputs at 4.5V and the output open.
8. Minimum required to guarantee a Write into the slowest bit.
9. Applied to the 8X300 based system with the data and address pins tied to the IV Bus.
10. SC + $\overline{ME}$ = 1 to avoid bus conflict.
11. WC + $\overline{ME}$ = 1 to avoid bus conflict.
12. The SC and WC outputs from the 8X300 are never at 1 simultaneously.

TIMING DEFINITIONS

T_{S1}	Required delay between beginning of Master Enable low and falling edge of Master Clock.
T_{SA}	Required delay between beginning of valid address and falling edge of Master Clock.
T_{HA}	Required delay between falling edge of Master Clock and end of valid Address.
T_{H1}	Required delay between falling edge of Master Clock and when Select Command becomes low.
T_{E1}	Delay between beginning of Select Command low and beginning of valid data output on the IV Bus.
T_{D1}	Delay between when select Command becomes high and end of valid data output on the IV Bus.
T_{H2}	Required delay between falling edge of Master Clock and when Master Enable becomes low.
T_{E2}	Delay between when Master Enable becomes low and beginning of valid data output on the IV Bus.
T_{D2}	Delay between when Master Enable becomes high and end of valid data output on the IV Bus.
T_{S2}	Required delay between when Select Command or Write Command becomes low and when Master Enable becomes low.
T_W	Minimum width of the Master Clock pulse.
T_{S3}	Required delay between when Master Enable becomes low and falling edge of Master Clock.
T_{H3}	Required delay between falling edge of Master Clock and when Master Enable becomes high.
T_{SD}	Required delay between beginning of valid data input on the IV Bus and falling edge of Master Clock.
T_{HD}	Required delay between falling edge of Master Clock and end of valid data input on the IV Bus.
T_{H4}	Required delay between falling edge of Master Clock and when Write Command becomes low.

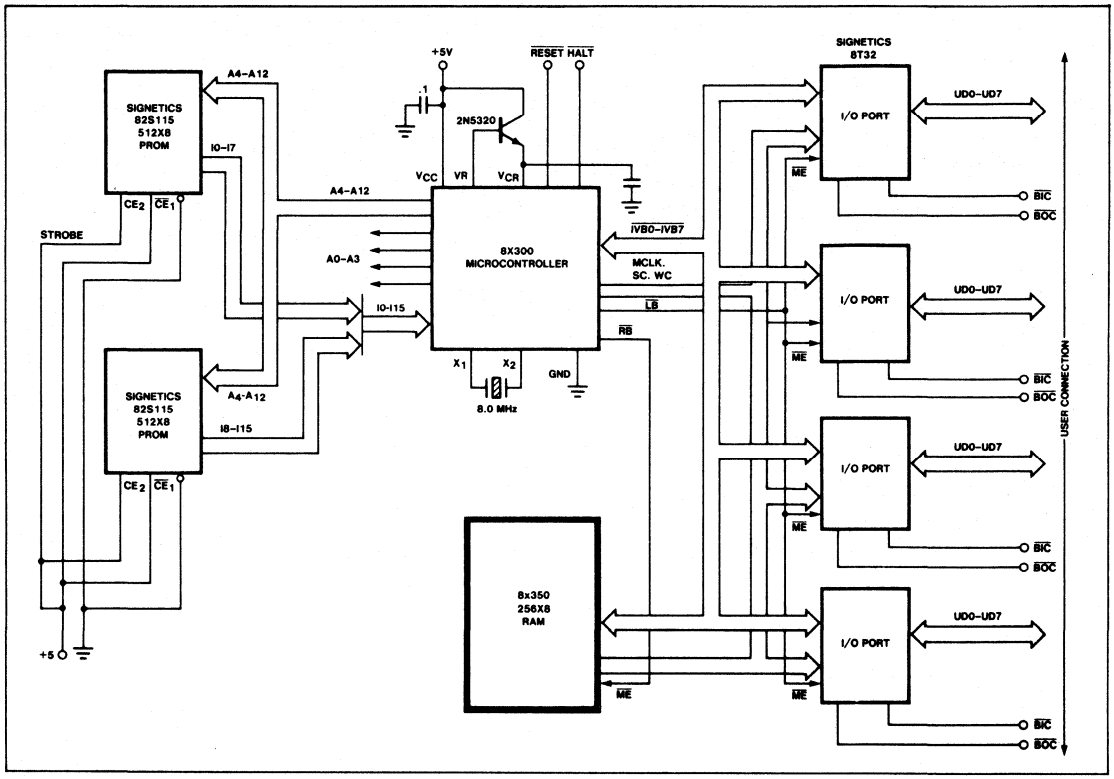
VOLTAGE WAVEFORM

INPUT PULSES

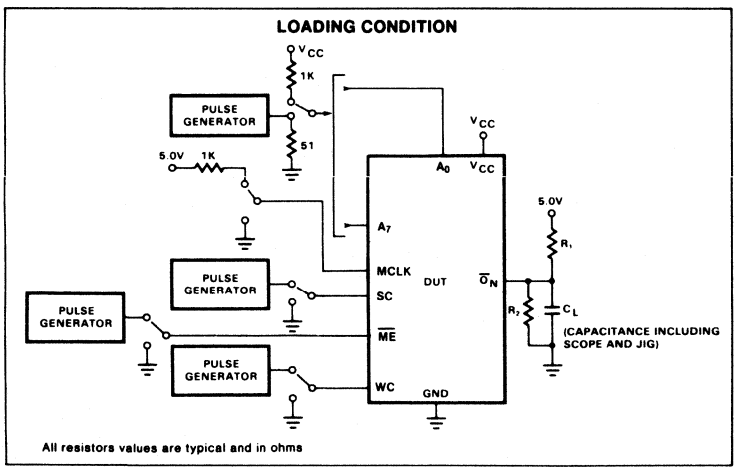


Measurements: All circuit delays are measured at the +1.5V level of inputs and output.

TYPICAL 8X350 APPLICATION



TEST LOAD CIRCUIT



Bipolar Ram (32X8)

Originally published by Signetics January 1985

Product Specification

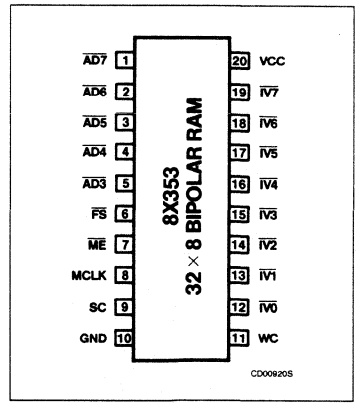
DESCRIPTION

The 8X353 is a 32-byte RAM designed principally as a working storage element in 8X305-based systems. The 8X353 is ideal for applications requiring a relatively small amount of data storage and maximum I/O flexibility. Since the 8X353 takes up only 32 of 256 locations on a controller bank, this device allows single cycle, bank-to-bank data transfer and other implementations in which the user does not wish to dedicate an entire I/O bank to data storage. Contributing to the versatility of the 8X353 is a fast select feature which, when supplemented with extended micro-code, allows this device to be selected externally of the IV bus.

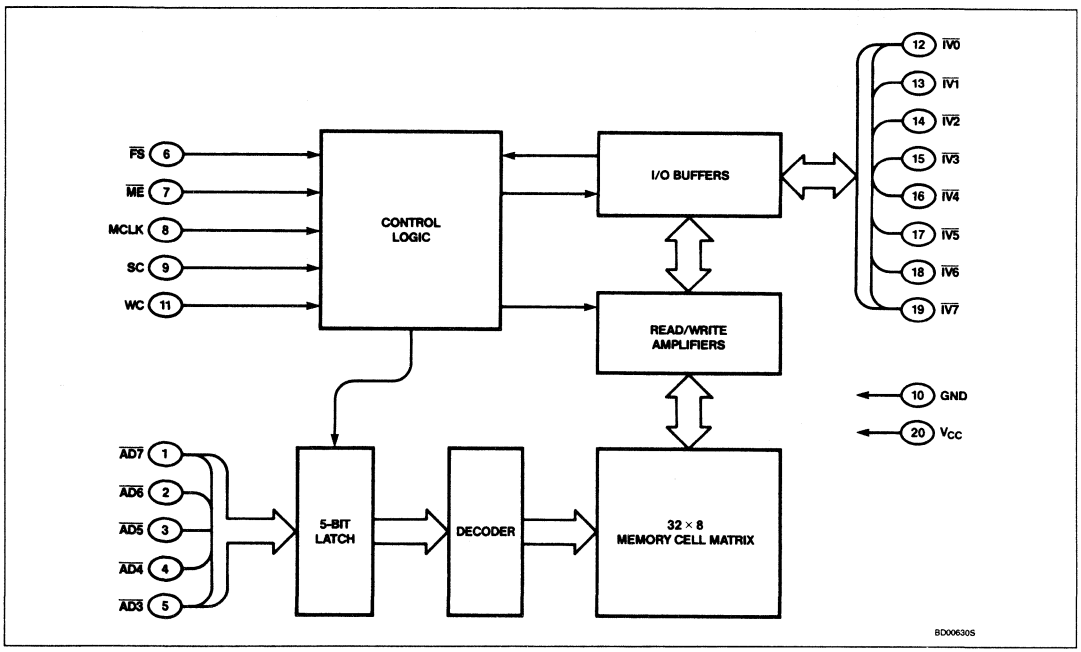
FEATURES

- **32 bytes of storage**
- **32 port addresses ($00_8 - 37_8$)**
- **224 bank addresses available for other 8X305 peripherals**
- **Fast Select feature for use with extended micro-code**
- **On-chip address decoding**
- **Separate address input pins**
- **Direct interface**
- **Single 5V power supply**
- **0.3 inch, Slim Line package**
- **0.3 inch, CERDIP package**

PIN CONFIGURATION

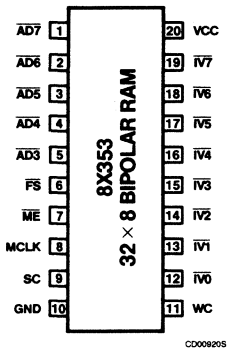


BLOCK DIAGRAM



Product Specification

PIN DESCRIPTION



PIN NO.	IDENTIFIER	FUNCTION
1 – 5	$\overline{AD7}$ – $\overline{AD3}$	ADdress bus – Active low inputs that define which memory locations are to be accessed.
6	$\overline{FS}$	Fast Select – When active low, this input allows read and write operations without performing a full select cycle.
7	$\overline{ME}$	Master Enable – When low, this input indicates the microcontroller bank is active; otherwise the chip will not respond to any signal on the IV bus.
8	MCLK	Master CLocK – A standard 8X305 clock input used for timing reference and system synchronization.
9	SC	Select Command – When active high, this input indicates IV bus data is to be interpreted as an address.
10	GND	GrouND.
11	WC	Write Command – When active high, this signal indicates data is being input from the IV bus.
12 – 19	$\overline{IV0}$ – $\overline{IV7}$	Interface Vector Bus – Three-state, active low, bidirectional signals which communicate I/O data and addresses (chip select).
20	V_{CC}	Supply Voltage.

Product Specification

FUNCTIONAL OPERATION

The 8X353 is capable of performing two functions — write and read. These operations, and the control logic necessary for their execution, are described in the following text and summarized in Table 1. Typical timing relationships for write, read, and select cycles are shown in Timing Diagrams.

Select Logic

Standard Select Cycle — Before a read or write can occur, the 8X353 must first be selected by either a standard select cycle or by Fast Select operation (see below). A standard select cycle is initiated when SC and MCLK are high, and $\overline{ME}$ and WC are low. The three most significant bits on the $\overline{IV}$ bus are then compared with fixed internal values (110_2 active low), and if identical, an internal select latch is activated (high) and the five bits of the memory address (pins AD7 – AD3) are loaded into an address latch.

As shown in Table 1, the 8X353 remains in the selected state until another select cycle is initiated, at which point the select latch and addresses are updated.

In situations where more than one RAM is required, the fixed internal values of IV0, IV1, and IV2 can be used as chip addresses. By scrambling these pins, up to three RAMs can be selected from a single IV bus.

Write/Read Operation

Write and read operations are possible only after the 8X353 has been selected by either a standard select cycle or by Fast Select operation. As shown in Table 1, data I/O is controlled by $\overline{ME}$, SC, WC, MCLK, FS, and the current state of the select latch.

After a data address has been latched, data is written to the specified memory location when MCLK and WC are high and $\overline{ME}$ is low; a read occurs when MCLK, WC, SC, and $\overline{ME}$ are low. In both cases, data is transmitted via the IV bus.

Table 1. Summary Of 8X353 Operation

MODE	CONDITIONS						RESULTS				
	$\overline{ME}$	SC	WC	MCLK	$\overline{FS}$	Select Latch	$\overline{IV}$ Bus	$\overline{AD}$ Bus	Data	Address Latch	Select Latch
A_{UF}	L	H	L	H	X	X	Input (chip address)	Input[2]	Keep	Update[2]	Update[2]
D_{IS}	L	L	H	H	H	H	Input	Ignore	Update	Keep	Keep
D_{OS}	L	L	L	L	H	H	Output	Ignore	Keep	Keep	Keep
A_{UF}	L	S	L	H	L	X	Ignore	Input	Keep	Update	Keep
A_{UF}	H	X	X	H	L	X	Ignore	Input	Keep	Update	Keep
D_{IF}	L	L	H	H	L	X	Input	Ignore	Update	Keep	Keep
D_{OF}	L	L	L	L	L	X	Output	Ignore	Keep	Keep	Keep
$\overline{ME}$ High	H	X	X	X	H	X	Ignore	Ignore	Keep	Keep	Keep
$\overline{ME}$ High	H	X	X	L	L	X	Ignore	Ignore	Keep	Keep	Keep
D_{OS} w/o MCLK Low	L	L	L	H	H	X	Ignore	Ignore	Keep	Keep	Keep
D_{OS} w/o select latch High	L	L	L	L	H	L	Ignore	Ignore	Keep	Keep	Keep
D_{IS} w/o MCLK High	L	L	H	L	X	X	Ignore	Ignore	Keep	Keep	Keep
D_{IS} w/o select latch High	L	L	H	H	H	L	Ignore	Ignore	Keep	Keep	Keep
A_{US} w/o MCLK High	L	H	L	L	X	X	Ignore	Ignore	Keep	Keep	Keep
Not defined	X	H	H	X	X	X			Not defined		

NOTES:
1. A_{US} = address update with standard select
   D_{IS} = data input with standard select
   D_{OS} = data output with standard select
   A_{UF} = address update with fast select
   D_{IF} = data input with fast select
   D_{OF} = data output with fast select
2. Depending on IV bus data
3. Fast select Logic ($\overline{FS}$) given but is not used for 8X305 operation.
   X = Don't care

Product Specification

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Power supply voltage	+7	V dc
V_{IN}	Input voltage	+5.5	V dc
V_O	Off-state voltage	+5.5	V dc
T_{STG}	Storage temperature range	−65 to +150	C

DC ELECTRICAL CHARACTERISTICS $V_{CC} = 5.0 \pm 5\%$, $0°C \leq T_A \leq 70°C$

PARAMETER		TEST CONDITIONS[1]	LIMITS			UNIT
			Min	Typ	Max	
V_{IH}	High level input voltage		2			V
V_{IL}	Low level input voltage				0.8	V
V_{OH}	High level output voltage	IV pins: V_{CC} = Min, $I_{OH} = -3.2mA$	2.4			V
V_{OL}	Low level output voltage	IV pins : V_{CC} = Min, I_{OL} = 16mA			0.55	V
V_{IC}	Input clamp voltage	V_{CC} = Min, $I_{IN} = -10mA$			−1.0	V
I_{IH}	High level input (or leakage) current	Address pins and control pins: V_{CC} = Max, V_{IN} = 4.5V			100	μA
I_{IL}	Low level input current	Address pins and control pins: V_{CC} = Max, V_{IN} = .5V			−100	μA
I_{OS}	Short circuit output current	IV pins: V_{CC} = Max[2]	−30		−140	mA
I_{OZH}	High-Z state output current – high level	IV pins: V_{CC} = Max, V_O = 2.7V			100	μA
I_{OZL}	High-Z state output current – low level	IV pins: V_{CC} = Max, V_O = .5V			−200	μA
I_{CC}	Supply current	All inputs: V_{CC} = Max, V_{IN} = 0V			200	mA

NOTES:
1. All voltages measured with respect to Ground terminal.
2. At any time, no more than one output should be connected to ground.

Product Specification

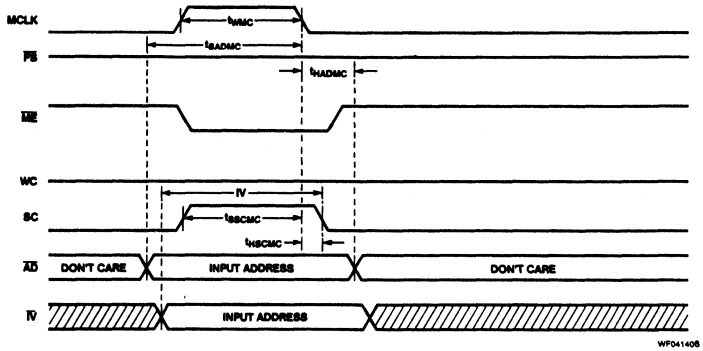
AC ELECTRICAL CHARACTERISTICS $V_{CC} = 5.0 \pm 5\%$, $0°C \leq T_A \leq 70°C$

PARAMETER	REFERENCES		TEST CONDITIONS[1,2,3]	LIMITS			UNIT
	From	To		Min	Typ	Max	
Pulse width							
t_{WMC} MCLK pulse width	↑MCLK	↓MCLK	FS = High;$\overline{ME}$ = Low	35			ns
Set-up times							
t_{SIVMC} IV set-up time	$\overline{IV}$	↓MCLK	FS = High;$\overline{ME}$ = Low	40			ns
t_{SADMC} Address set-up time	$\overline{AD}$	↓MCLK	FS, SC = High; WC,$\overline{ME}$ = Low	40			ns
t_{SWCMC} Write command set-up time	↑WC	↓MCLK	FS = High; SC,$\overline{ME}$ = Low	60			ns
t_{SSCMC} Select command set-up time	↑SC	↓MCLK	FS = High; WC,$\overline{ME}$ = Low	55			ns
t_{SMEMC} Master enable set-up time	↓$\overline{ME}$	↓MCLK	FS = High	55			ns
Hold times							
t_{HIVMC} IV hold time	↓MCLK	$\overline{IV}$	FS = High;$\overline{ME}$ = Low	5			ns
t_{HADMC} Address hold time	↓MCLK	$\overline{AD}$	FS, SC = High; WC,$\overline{ME}$ = Low	5			ns
t_{HWCMC} Write command hold time	↓MCLK	↓WC	FS = High; SC,$\overline{ME}$ = Low	3			ns
t_{HSCMC} Select command hold time	↓MCLK	↓SC	FS = High; WC,$\overline{ME}$ = Low	0			ns
t_{HMEMC} Master enable hold time	↓MCLK	↑$\overline{ME}$	FS = High; SC, WC = Low	5			ns
Output enable times							
t_{EDOMC} IV output enable time	↓MCLK	$\overline{IV}$	FS = High; WC,$\overline{ME}$, SC = Low			30	ns
t_{EDOWC} IV output enable time	↓WC	$\overline{IV}$	FS = High; MCLK,$\overline{ME}$, SC = Low			30	ns
t_{EDOSC} IV output enable time	↓SC	$\overline{IV}$	FS = High; MCLK, WC,$\overline{ME}$ = Low			30	ns
t_{EDOME} IV output enable time	↓$\overline{ME}$	$\overline{IV}$	FS = High; MCLK, WC, SC = Low			30	ns
Output disable times							
t_{DDOMC} IV output disable time	↑MCLK	$\overline{IV}$	FS = High; WC,$\overline{ME}$, SC = Low[4]			25	ns
t_{DDOWC} IV output disable time	↑WC	$\overline{IV}$	FS = High; MCLK,$\overline{ME}$ = Low[4]			25	ns
t_{DDOSC} IV output disable time	↑SC	$\overline{IV}$	FS = High; MCLK, WC,$\overline{ME}$ = Low[4]			25	ns
t_{DDOME} IV output disable time	↑$\overline{ME}$	$\overline{IV}$	FS = High; MCLK, WC = Low[4]			25	ns

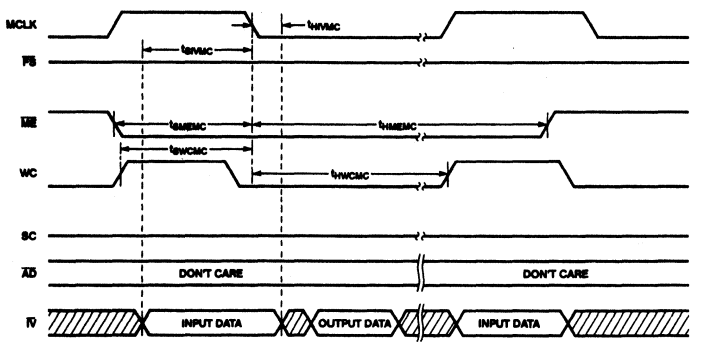
NOTES:
1. Loading: see Test Loading Circuits
2. All inputs are driven between 0 and 3.0 volts.
3. Except for output disable times, all timing parameters are measured at 1.5 volts.
4. These parameters are measured with a capacitive loading of 50pF and represent the output driver turn-off time. Disable times measured into a capacitive loading of 300pF will be a maximum of 40ns.

Product Specification

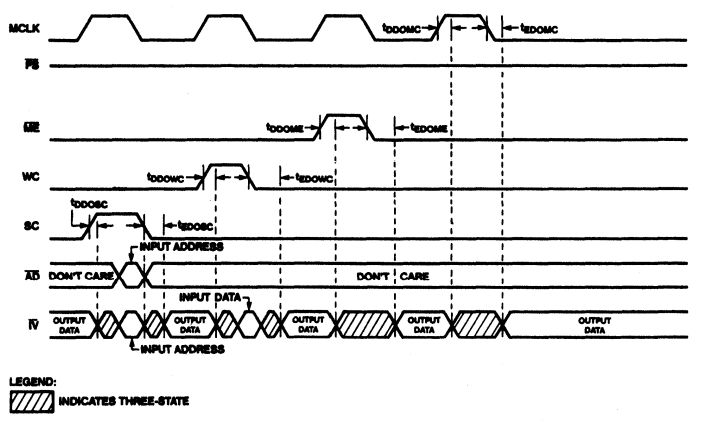
TIMING DIAGRAM



a. Standard Select Cycle Timing



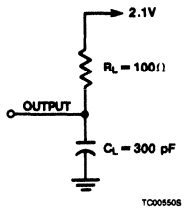
b. Write/Read Cycle Timing



c. Output Enable/Disable Timing

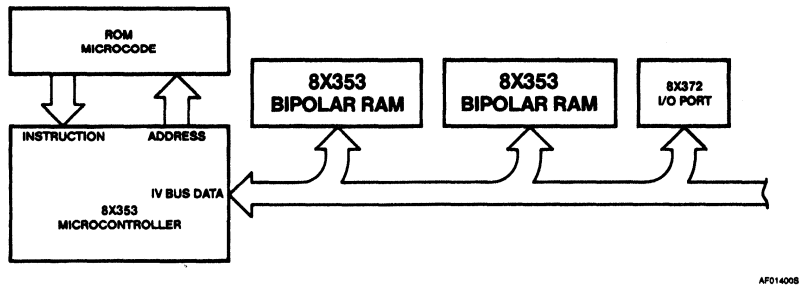
Product Specification

TEST LOADING CIRCUIT

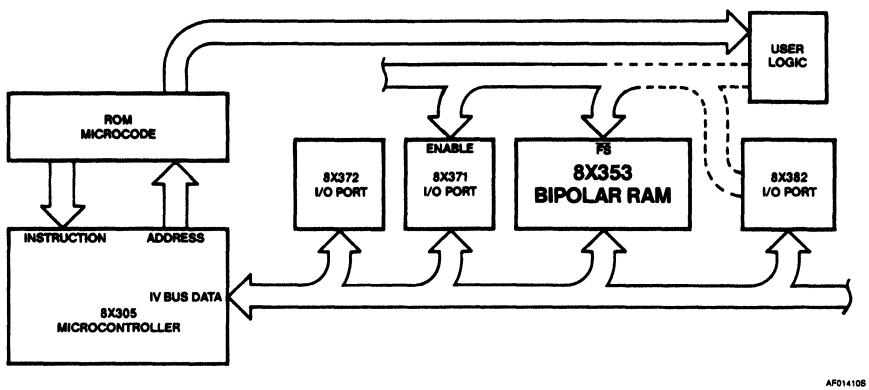


NOTE:
C_L = 50pF for output disable times.

APPLICATIONS DIAGRAMS



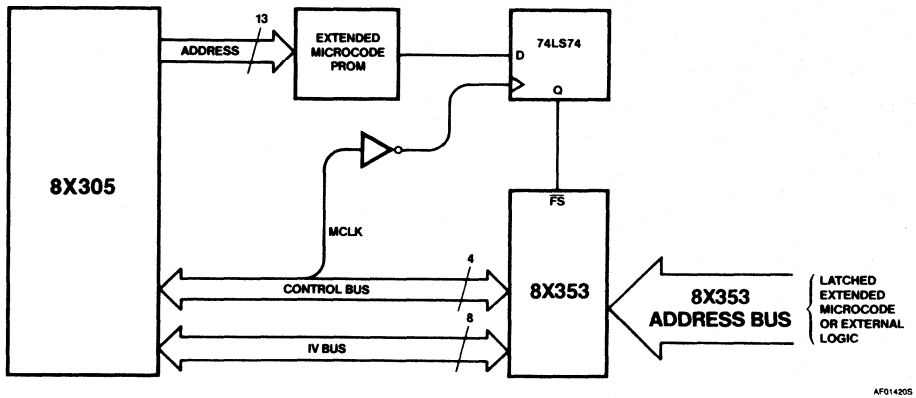
a. Implementation of the 8X353 Using $\overline{IV}$ Bus Select



b. Implementation of the 8X353 Using FAST Select and External Logic

Product Specification

APPLICATIONS DIAGRAM (Continued)



NOTES:
1. The control bus consists of MCLK, SC, WC and $\overline{ME}$.
2. The 8X353 can be selected using the $\overline{FS}$ pin instead of a normal select cycle. Since the flip-flop is clocked by the falling edge of MCLK, the data input to the flip-flop must have sufficient set-up time.

Example Use of the 8X353 Fast Select Mode.

ORDERING INFORMATION

N8X353N (Plastic)
N8X353F (CERDIP)

LIFO STACK MEMORY (32x8)

Originally published by Signetics January 1984

FEATURES

- **32 bytes of storage**
- **Cascadable LIFO operation**
- **Dedicated port address**
- **Fast select feature for use with extended microcode**
- **Three state TTL outputs**
- **Single 5 volt power supply**
- **0.3 inch, slim line package**

PRODUCT DESCRIPTION

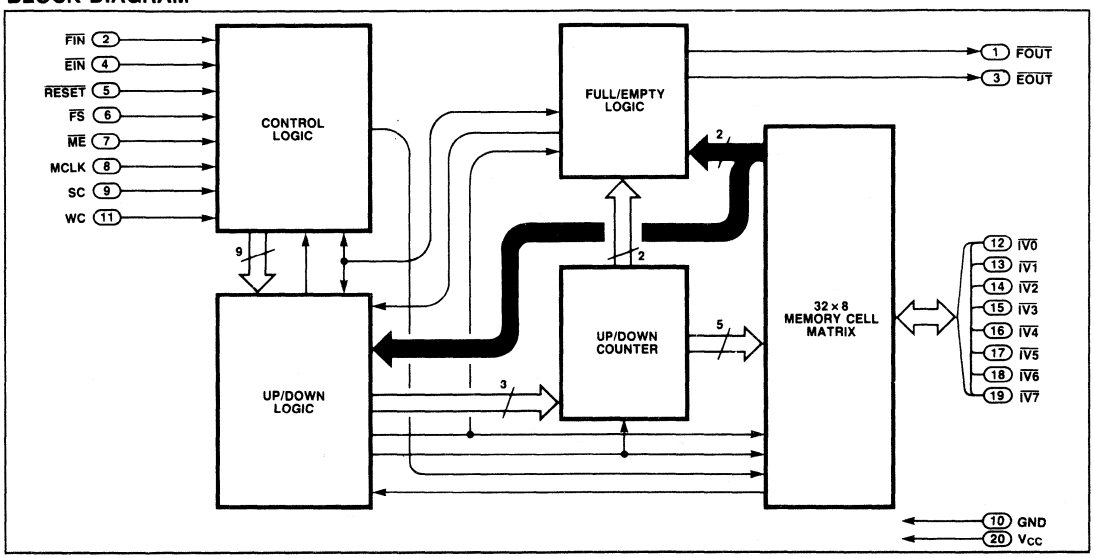
The 8X355 is a Last In/First Out stack memory designed to be compatible with 8X305-based systems. In addition to a 32-byte storage capacity, the 8X355 contains the necessary logic to allow cascaded operation.

The LIFO stack may be addressed using either conventional 8X305 techniques, i.e., $\overline{IV}$ bus select, or, in systems where extremely high performance is required, extended microcode can be implemented. A single enabling address is employed so that once enabled, a stack of 8X355s can accept an uninterrupted stream of data. By omitting the need for address select cycles prior to each data access, the LIFO stack delivers a much higher performance than conventional memories; this feature is particularly valuable for saving internal registers during interrupt servicing.

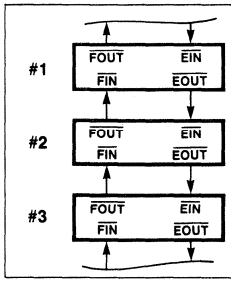
ORDERING INFORMATION

Contact Local Sales Representative

BLOCK DIAGRAM



TYPICAL CASCADED CONFIGURATION



	INPUT				RESULTING OUTPUT				COMMENTS
	$\overline{EIN}$	$\overline{FIN}$	$\overline{EOUT}$	$\overline{FOUT}$	$\overline{EOUT}$	$\overline{FOUT}$	PUSH	POP	
#1	X	X	L	L	Not possible				Empty and full status
	L	L	L	H	Update	H	Yes	No	Top of data stack at top of #1
#2	L	L	H	L	H	Update	No	Yes	Top of data stack at top of #2
	L	L	H	H	Update	Update	Yes	Yes	Top of data stack within #2
	X	H	X	X	L	H	No	No	Top of data stack below #2
#3	H	L	X	H	Not defined				Top of data stack moves from bottom to top
	H	L	H	L	H	L	No	No	Top of data stack above #2

Notes:
Status signals and Push/Pop operations reference stack chip #2
X = don't care

MEMORY ADDRESS DIRECTOR

Originally published by Signetics January 1984

FEATURES

- **Address control for working storage**
- **16-bit addressing capability**
- **Byte and word addressing support**
- **Automatic increment and decrement**
- **11 Address and word-count registers**
- **Reduces number of 8X305 instructions required**

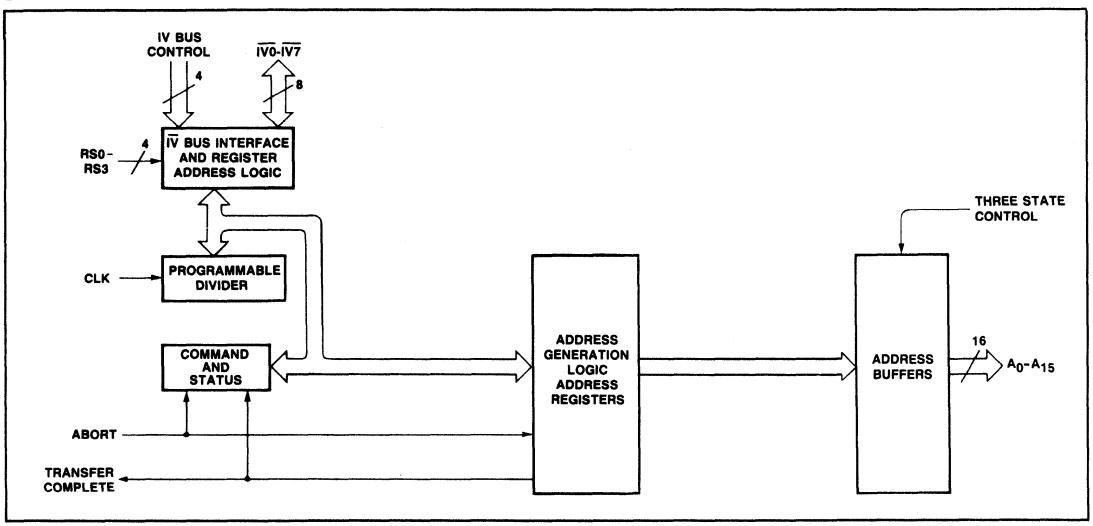
PRODUCT DESCRIPTION

The 8X360 Memory Address Director (MAD) is a high-performance member of the 8X300 Family that generates sequential memory addresses to facilitate the transfer of data to and from memory. The MAD provides a highly-efficient and cost-effective solution for DMA and other applications requiring large working-storage memories and high-speed data transfers. Once initialized with such information as starting address, ending address, byte count, address increment, address decrement, etc., the 8X360 performs all bookkeeping chores automatically and all address-management software is off-loaded from the processor. The 8X360 can be addressed by conventional means or by extended microcode; system status is available to the user via I/O pins.

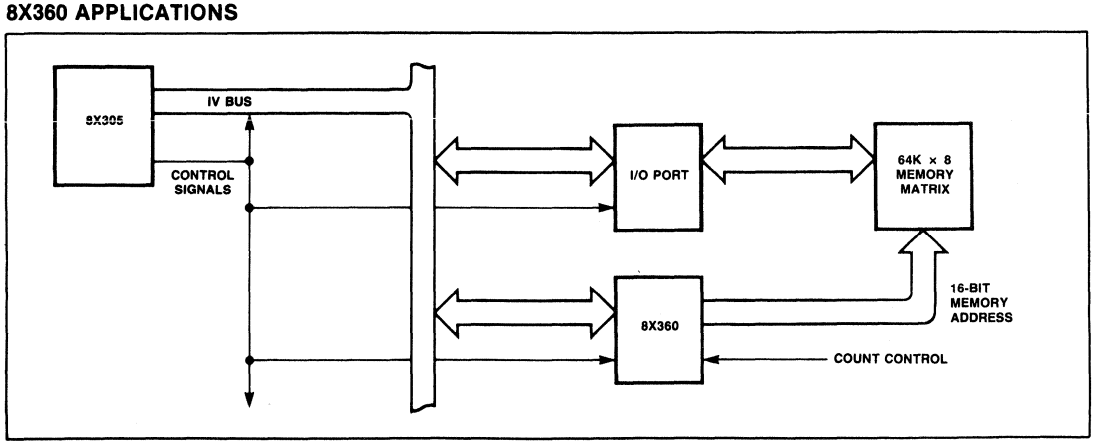
ORDERING INFORMATION

N8X360N, N8X360I, S8X360I

BLOCK DIAGRAM OF 8X360



8X360 APPLICATIONS



8-BIT LATCHED BIDIRECTIONAL I/O PORT

Originally published by Signetics January 1984

FEATURES

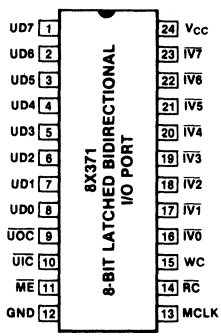
- **Two bidirectional 8-bit busses**
- **Independent bus operation (user-bus priority for data entry)**
- **User data input synchronous with respect to MCLK**
- **Three-state TTL outputs with high-drive capabilities**
- **Power-up to predetermined logic state**
- **Directly compatible with 8X305 (or 8X300) MicroControllers**
- **Single +5V supply**
- **0.4 inch 24-pin DIP**

PRODUCT DESCRIPTION

The 8X371 I/O Port is a bidirectional device designed for use as an interface element in systems that use TTL-compatible busses. Typically, the 8X371 is used with the 8X305 MicroController and its associated Interface Vector ($\overline{IV}$) bus; however, it can also be used with the 8X300 MicroController or an equivalent microprocessor. The 8X371 is functionally the same and pin-for-pin compatible with the older 8T31/8X31 but features improved performance and increased drive current. As shown in the logic diagram of Figure 1, the 8X371 consists of eight identical data latches—bits 0 through 7. The latches are accessed from either of two 8-bit busses—the MicroController ($\overline{IV}$ bus) and the user data (UD bus). Separate controls are provided for each bus and both busses operate independently, except when both attempt to input data at the same time; in such situations, the user bus always has priority. A Master Enable ($\overline{ME}$) input is available for additional control over the $\overline{IV}$ bus. The data latches are transparent, in that, while either bus is enabled for input, all input-data transitions are propagated to the other bus, if enabled for output.

8X371 PACKAGE and PIN DESIGNATIONS

N, I PACKAGE



TOP VIEW

ORDER NUMBERS

N8X371N, N8X371I
S8X371I/883B, S8X371I/883C

PIN. NO.	IDENTIFIER	FUNCTION
1-8	UD7-UD0	Three-state, bidirectional User Data (UD) bus; UD0 and corresponds to $\overline{IV0}$.
9	$\overline{UOC}$	User Output Control—active low input to enable data output to UD0-UD7.
10	$\overline{UIC}$	User Input Control—active low input to enable data input from UD0-UD7.
11	$\overline{ME}$	Master Enable—active low input to enable the $\overline{IV}$ bus for data input, or data output; UD-bus operations are unaffected.
12	GND	Ground.
13	MCLK	Master Clock—active high input (from MicroController used to strobe data into data latches from the $\overline{IV}$ and UD buses.
14	$\overline{RC}$	Read Control—active low input to enable data output to $\overline{IV0}$-$\overline{IV7}$.
15	WC	Write Command—active high input (from MicroController) to enable the writing of data into the data latches from the $\overline{IV}$ bus (provided $\overline{UIC}$ is not low).
16-23	$\overline{IV0}$-$\overline{IV7}$	Interface Vector (Input/Output Bus — three-state, bidirectional, MicroController data bus; $\overline{IV0}$ corresponds to UD0.
24	VCC	+5V power supply.

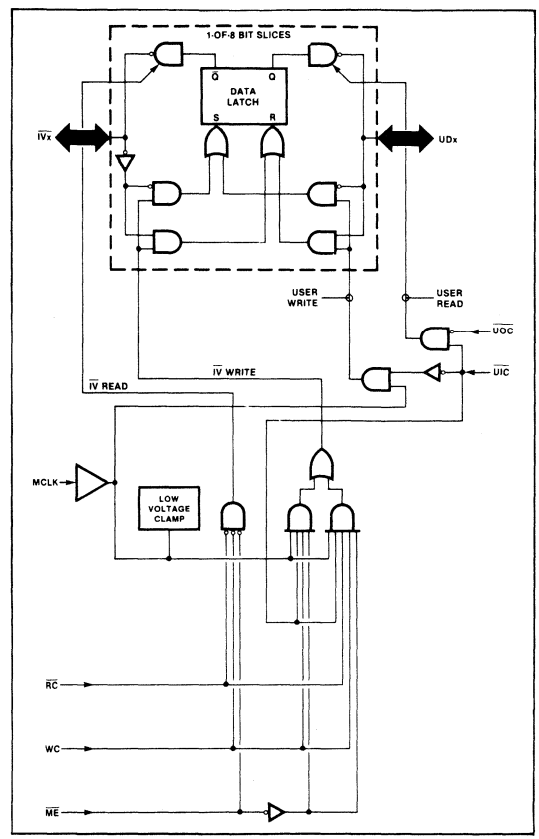


Figure 1. Logic Diagram for 8X371 I/O Port

FUNCTIONAL OPERATION

UD Bus Control

As shown in Table 1, the User Data (UD) bus interface is controlled by the $\overline{UIC}$ and $\overline{UOC}$ inputs. Data input to the UD bus is synchronous with MCLK, that is, with $\overline{UIC}$ low, information is written into the data latches only when MCLK is high. Output drivers on the UD bus are enabled when $\overline{UOC}$ is low and $\overline{UIC}$ is high.

Table 1. INPUT/OUTPUT CONTROL OF UD BUS

$\overline{UIC}$	$\overline{UOC}$	MCLK	FUNCTION OF UD BUS
H	L	X	Output data
L	X	H	Input data
L	X	L	Inactive
H	H	X	Inactive

X = don't care

$\overline{IV}$ Bus Control

Input/output control of the $\overline{IV}$ bus is shown in Table 2; this bus is controlled by $\overline{RC}$, WC, $\overline{ME}$, and MCLK. The $\overline{IV}$ bus is enabled for output (MicroController read operation) when $\overline{ME}$, $\overline{RC}$, and WC are all low. Data is written into the data latches from the $\overline{IV}$ bus when $\overline{ME}$ is low and both WC and MCLK are high. To avoid data-input conflicts, inputs from the $\overline{IV}$ bus are inhibited when $\overline{UIC}$ is low; under all other conditions, the $\overline{IV}$ and UD busses operate independently. The MicroController Left Bank ($\overline{LB}$) and Right Bank ($\overline{RB}$)

Table 2. INPUT/OUTPUT CONTROL OF IV BUS

$\overline{ME}$	$\overline{RC}$	WC	MCLK	$\overline{UIC}$	FUNCTION OF $\overline{IV}$ BUS
L	L	L	X	X	Output Data
L	X	H	H	H	Input Data
L	H	L	X	X	Inactive
L	X	H	X	L	Inactive
L	X	H	L	H	Inactive
H	X	X	X	X	Inactive

outputs can control the $\overline{ME}$ inputs for two banks of I/O devices, thus acting as a ninth address bit. If more than one I/O Port (including the addressable parts—8X372, 8X376, 8X382, etc.) are to be connected to the same bank ($\overline{LB}$ or $\overline{RB}$) of the MicroController, selection of each 8X371 must be accomplished with external control logic to avoid bus conflicts.

Bus Logic Levels

Data written into the I/O port from either bus will appear inverted when read from the other bus. Data written into either bus will not be inverted when read from the same bus. (Note. *A logic "1" in MicroController software corresponds to a high level on the UD bus even though the $\overline{IV}$ bus is inverted.*) The 8X382 wakes up in the unselected state with all data bits latched at the "logic 1" level (UD bus outputs high if enabled).

DC ELECTRICAL CHARACTERISTICS

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Power supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
T_{STG}	Storage temperature range	-65 to +150	°C

COMMERCIAL: $4.75V \leq V_{CC} \leq 5.25V$, $0°C \leq T_A \leq 70°C$
MILITARY: $4.5V \leq V_{CC} \leq 5.5V$, $-55°C \leq T_C \leq 125°C$

PARAMETER		TEST CONDITIONS	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	Supply Voltage		4.75	5	5.25	4.5	5	5.5	V
V_{IH}	High Level Input Voltage		2.0			2.0			V
V_{IL}	Low Level Input Voltage				0.8			0.8	V
V_{CL}	Input Clamp Voltage	V_{CC}=Min; I_I=-10mA			-1.5			-1.5	V
I_{IH}	High Level Input Current[1]	V_{CC}=Max; V_{IH}=2.7V		5	100		5	100	μA
I_{IL}	Low Level Input Current[1]	V_{CC}=Max; V_{IL}<0.5V		-350	-550		-350	-550	μA
V_{OL}	Low Level Output Voltage $\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC}=Min; I_{OL}= 16mA			0.55			0.55	V
	User Bus (UD4-UD7)	V_{CC}=Min; I_{OL}=24mA			0.55			0.55	V
V_{OH}	High Level Output Voltage	V_{CC}=Min; I_{OH}=-3.2mA	2.4			2.4			V
I_{OS}	Short Circuit Output Current[3] $\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC}=Max	-20			-20			mA
	UD Bus (UD4-UD7)	V_{CC}=Max	-10			-10			mA
I_{CC}	Supply Current	V_{CC}=Max; $\overline{ME}$=$\overline{UOC}$=V_{CC}		90	150		90	150	mA

Notes:
1. The input current includes the Three-state leakage current of the output driver on the data lines.
2. Only one output may be shorted at a time.

AC ELECTRICAL CHARACTERISTICS (Cont'd)

COMMERCIAL: $4.75 \le V_{CC} \le 5.25V$, $0°C \le T_A \le 70°C$

MILITARY: $4.5V \le V_{CC} \le 5.5V$, $-55°C \le T_C \le 125°C$

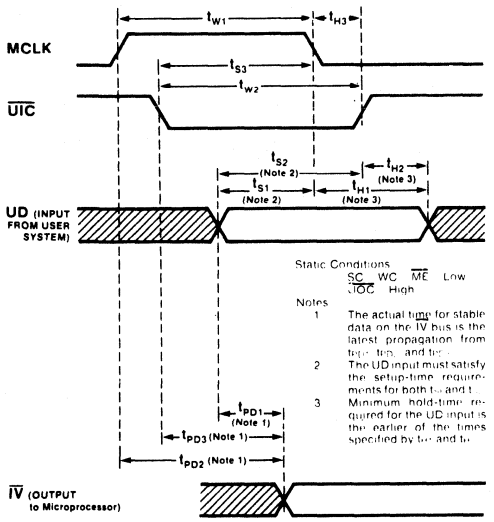
LOADING: See TEST LOADING CIRCUITS

PARAMETER		REFERENCES		TEST CONDITIONS	LIMITS (Commercial)			LIMITS (Military)			UNIT
		FROM	TO		Min	Typ	Max	Min	Typ	Max	
Pulse Widths:											
t_{W1}	Clock High	↑MCLK	↓MCLK		35			35			ns
t_{W2}	User Input Control	↓$\overline{UIC}$	↑$\overline{UIC}$	MCLK = High	35			35			ns
Propagation Delays:											
t_{PD1}	UD Propagation Delay	UD	$\overline{IV}$	MCLK = High $\overline{RC}$ = WC = $\overline{ME}$ = $\overline{UCI}$ = Low			30			30	ns
t_{PD2}	UD Clock Delay	↑MCLK	$\overline{IV}$	UD = Stable; $\overline{RC}$ = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			50			50	ns
t_{PD3}	UD Input Delay	↓$\overline{UIC}$	$\overline{IV}$	UD = Stable; MCLK = High $\overline{RC}$ = WC = $\overline{ME}$ = Low			50			50	ns
t_{PD4}	$\overline{IV}$ Data Propagation Delay	$\overline{IV}$	UD	MCLK = WC = $\overline{UIC}$ = High; $\overline{ME}$ = $\overline{UOC}$ = $\overline{RC}$ = Low			45			45	ns
t_{PD5}	$\overline{IV}$ Data Clock Delay	↑MCLK	UD	WC = $\overline{UIC}$ = High; $\overline{IV}$ = Stable $\overline{ME}$ = $\overline{UOC}$ = $\overline{RC}$ = Low			55			55	ns
Output Enable Timing:											
t_{OE1}	UD Output Enable	↓UOC	UD	$\overline{UIC}$ = High			30			30	ns
t_{OE2}	UD Input Recovey	↑$\overline{UIC}$	UD	$\overline{UOC}$ = Low			30			30	ns
t_{OE3}	$\overline{IV}$ Data Master Enable	↓$\overline{ME}$	$\overline{IV}$	WC = $\overline{RC}$ = Low			22			25	ns
t_{OE4}	$\overline{IV}$ Data Read Enable	↓$\overline{RC}$	$\overline{IV}$	WC = $\overline{ME}$ = Low			25			25	ns
t_{OE5}	$\overline{IV}$ Data Write Recovery	↓WC	$\overline{IV}$	$\overline{RC}$ = $\overline{ME}$ = Low			25			25	ns
Output Disable Timing:											
t_{OD1}	UD Output Disable	↑$\overline{UOC}$	UD	$\overline{UIC}$ = High			25			25	ns
t_{OD2}	UD Input Override	↓$\overline{UIC}$	UD	$\overline{UOC}$ = Low			30			30	ns
t_{OD3}[1]	$\overline{IV}$ Data Master Disable	↑$\overline{ME}$	$\overline{IV}$	WC = $\overline{RC}$ = Low			20			20	ns
t_{OD4}[1]	$\overline{IV}$ Data Read Disable	↑$\overline{RC}$	$\overline{IV}$	WC = $\overline{ME}$ = Low			20			20	ns
t_{OD5}	$\overline{IV}$ Data Write Override	↑WC	$\overline{IV}$	$\overline{RC}$ = $\overline{ME}$ = Low			20			20	ns
Setup Time:											
t_{S1}	UD Clock Setup Time	UD	↓MCLK	$\overline{UIC}$ = Low	15			15			ns
t_{S2}	UD Setup Time	UD	↑UIC	MCLK = High	15			15			ns
t_{S3}	User Input Control Setup Time	↓$\overline{UIC}$	↓MCLK		25			25			ns
t_{S4}	$\overline{IV}$ Data Setup Time	$\overline{IV}$	↓MCLK	WC = $\overline{UIC}$ = High; $\overline{ME}$ = Low	35			35			ns
t_{S5}[2]	$\overline{IV}$ Master Enable Setup Time	↓$\overline{ME}$	↓MCLK	WC = $\overline{UIC}$ = High	30			30			ns
t_{S6}	$\overline{IV}$ Write Control Setup Time	↑WC	↓MCLK	$\overline{ME}$ = Low; $\overline{UIC}$ = High	30			30			ns

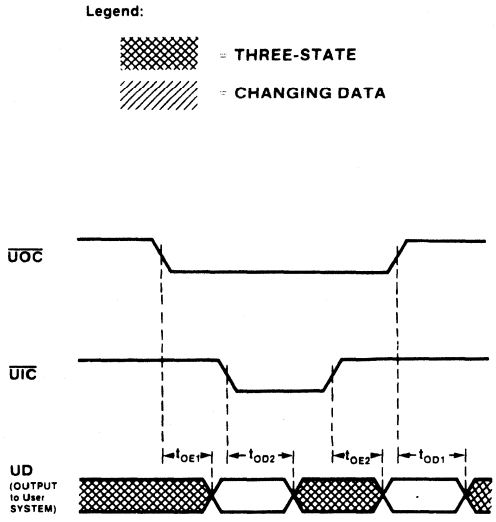
AC ELECTRICAL CHARACTERISTICS (Cont'd)

PARAMETER	REFERENCES		TEST CONDITIONS	LIMITS (Commercial)			LIMITS (Military)			UNIT
	FROM	TO		Min	Typ	Max	Min	Typ	Max	
Hold Times:										
t_{H1} UD Clock Hold Time	↓MCLK	UD	$\overline{UIC}$ = Low	15			15			ns
t_{H2} UD Control Hold Time	↑$\overline{UIC}$	UD	MCLK = High	15			15			ns
t_{H3} User Input Control Hold Time	↓MCLK	↑UIC		0			0			ns
t_{H4} $\overline{IV}$ Data Hold Time	↓MCLK	$\overline{IV}$	WC = $\overline{UIC}$ = High; $\overline{ME}$ = Low	5			5			ns
t_{H5}[2] $\overline{IV}$ Master Enable Hold Time	↓MCLK	↑$\overline{ME}$	WE = $\overline{UIC}$ = High	0			0			ns
t_{H6} $\overline{IV}$ Write Control Hold Time	↓MCLK	↓WC	$\overline{ME}$ = Low; = $\overline{UIC}$ = High	0			0			ns

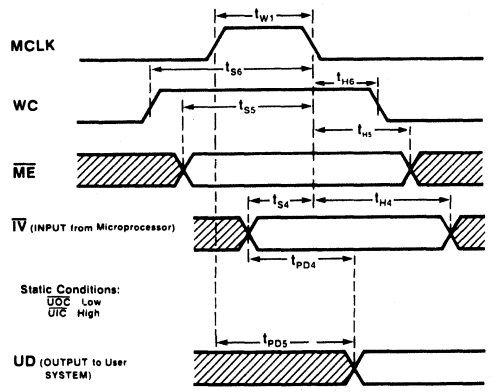
Notes:
1. These parameters are measured with a capacitive loading of 50 *pf* and represent the output driver turn-off time.
2. If $\overline{ME}$ is to be high (inactive), it must be setup before the rising edge and held after the falling edge of MCLK to avoid unintended writing into or selection of the I/O port.



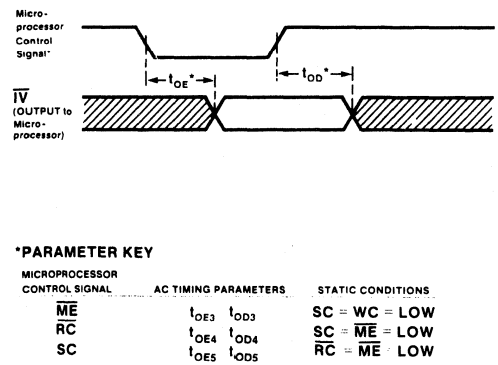
a. User Data Input Timing



b. User Data Output Timing



c. MicroController Write Cycle Timing



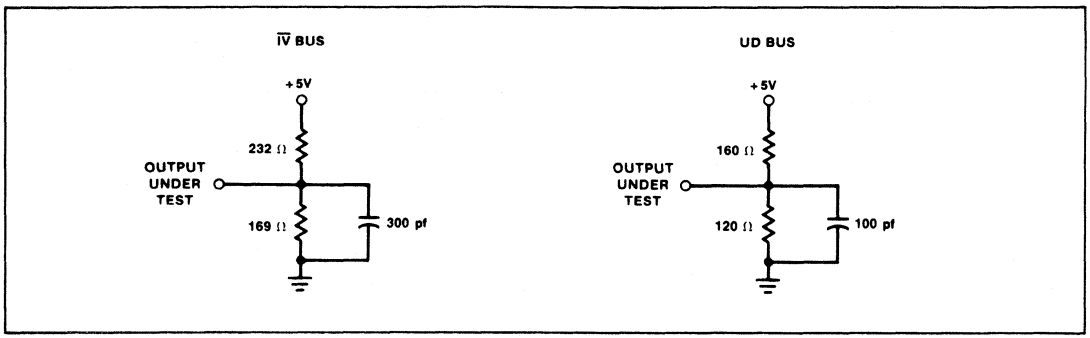
*PARAMETER KEY

MICROPROCESSOR CONTROL SIGNAL	AC TIMING PARAMETERS	STATIC CONDITIONS
ME	t_{OE3} t_{OD3}	SC = WC = LOW
RC	t_{OE4} t_{OD4}	SC = ME = LOW
SC	t_{OE5} t_{OD5}	RC = ME = LOW

d. MicroController Select Cycle Timing

Figure 2. Timing Diagram

TEST LOADING CIRCUITS



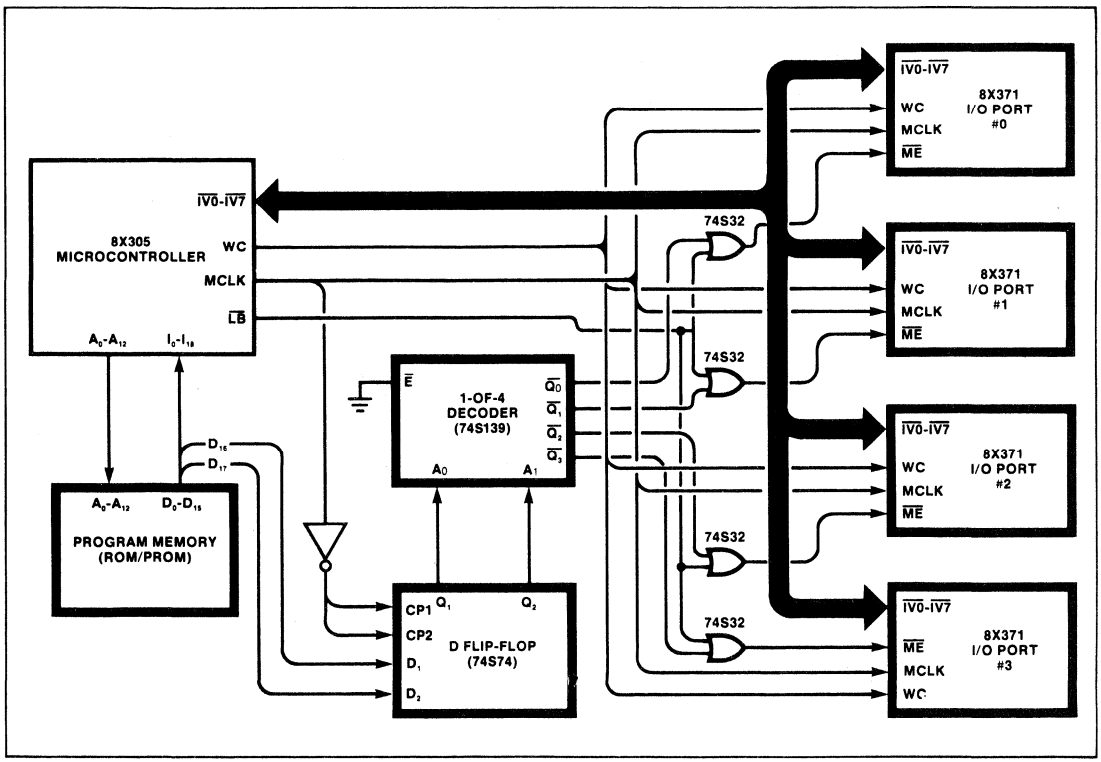
APPLICATIONS

In some applications, performance of a MicroController system can be enhanced by using the 8X371 I/O Port instead of an addressable 8X372 port. Using a technique referred to as Extended Microcode or Fast $\overline{IV}$ Select, the address select cycles which normally precede a read or write operation when using an 8X372 can be eliminated by use of the 8X371.

This technique is often used in bit slice microprocessor designs and involves widening the program memory beyond the normal 16-bit requirement of the MicroController. The extra bits are used as enable signals for the 8X371 ports. Thus, the 8X371 is enabled during the instruction cycle in which it is required for input/output operations. Since the software overhead of separate address select cycles is eliminated, the overall system performance is improved.

As shown in the accompanying diagram, the program memory is extended by two bit positions (D_{16} and D_{17}), permitting any one of four 8X371 ports to be enabled during those instructions that perform input/output operations. Because of timing considerations, latches must be used to hold the Extended Microcode through the end of the instruction cycle. A decoder is used to obtain four enable signals from the two extra bits. The decoder outputs are ORed with the $\overline{LB}$ output of the 8X305; thus, all four I/O ports are placed on the Left Bank of the $\overline{IV}$ bus.

I/O PORT SELECTION USING EXTENDED MICROCODE



ADDRESSABLE/BIDIRECTIONAL I/O PORTS

Originally published by Signetics January 1984

FEATURES

- **Two bidirectional 8-bit busses**
- **Independent bus operation (user-bus priority for data entry)**
- **User data input synchronous (8X372) or asynchronous (8X376) with respect to MCLK**
- **Programmed MicroController port address**
- **Three-state TTL outputs with high-drive capabilities**
- **Power-up to predetermined logic state**
- **Directly compatible with 8X305 or 8X300 MicroControllers**
- **Single +5V supply**
- **0.4 inch 24-pin DIP**

PRODUCT IDENTITY

8X372—Synchronous, three-state, bidirectional I/O port with programmed address.
8X376—Asynchronous, three-state, bidirectional I/O port with programmed address.

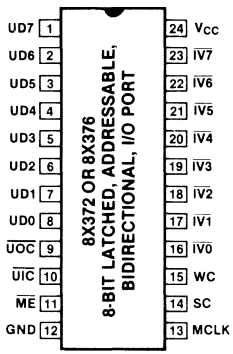
PRODUCT DESCRIPTION

Each of these I/O ports is an addressable device designed for use as a bidirectional interface element in systems that use TTL-compatible busses. Typically, these I/O ports are used with the 8X305 MicroController and its associated Interface Vector ($\overline{IV}$) bus; however, either port can also be used with the 8X300 MicroController or an equivalent microprocessor. The 8X372 and 8X376 are funtionally the same and pin-for-pin compatible with their respective counterparts, the 8T32/8X32 and 8T36/8X36; however, the new parts feature better performance, increased drive current, and improved programming procedures.

As shown in the logic diagram of Figure 1, each I/O port consists of eight identical data latches—bits 0 through 7. These latches are accessed through either of two 8-bit busses—one connecting to the MicroController ($\overline{IV}$ bus) and the other to the user system (UD bus). Separate controls are provided for each bus and both busses operate independently, except when both attempt to input data at the same time. In such situations, the user bus always has priority. The data latches are transparent, in that, while either bus is enabled for input, all transitions in input data are propagated to the other bus, if enabled for output.

Both the 8X372 and 8X376 are available with preprogrammed addresses (0_{10} through 255_{10}); either device can be field-programmed over the same address range. Input/output operations can begin once the I/O port is selected and appropriate control signals are generated. Port selection is implemented by putting the I/O port address (0_{10}-255_{10}) on the $\overline{IV}$ bus; once selected, the I/O port remains selected until a different "port address" is put on the bus. Thus, software overhead is minimized. Data is accessible on the UD bus at all times. A Master Enable ($\overline{ME}$) input, which is typically connected to the Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) output of the MicroController, provides the capability of organizing the $\overline{IV}$ bus into two separate and independent banks of I/O devices.

N, I PACKAGE



TOP VIEW

ORDER NUMBERS

N8X372N or N8X376N
N8X372I or N8X376I
S8X372I/883B or S8X372I/883C
S8X376I/883B or S8X376I/883C

PIN. NO.	IDENTIFIER	FUNCTION
1-8	UD7-UD0	Three-state, bidirectional User Data (UD) bus; UD0 corresponds to $\overline{IV0}$
9	$\overline{UOC}$	User Output Control – active low input to enable data output to UD0-UD7
10	$\overline{UIC}$	User Input Control – active low input to enable data input from UD0-UD7
11	$\overline{ME}$	Master Enable – active low input to enable the $\overline{IV}$ bus for data input, data output, or $\overline{IV}$ address selection deselection; UD-bus operations are unaffected
12	GND	Ground
13	MCLK	Master Clock – active high input from MicroController used to strobe data into data latches from the $\overline{IV}$ bus and, for the synchronous 8X372, from the UD bus. MCLK also synchronizes $\overline{IV}$ address selection
14	SC	Select Command – active high input from MicroController to enable $\overline{IV}$ address input from the $\overline{IV}$ bus for device selection
15	WC	Write Command – active high input from MicroController to enable the writing of data into the data latches from the $\overline{IV}$ bus, provided $\overline{UIC}$ is not low
16-23	$\overline{IV0}$-$\overline{IV7}$	Interface Vector Input Output Bus – three-state, bidirectional, MicroController data bus; $\overline{IV0}$ corresponds to UD0
24	V_{CC}	+5V power supply

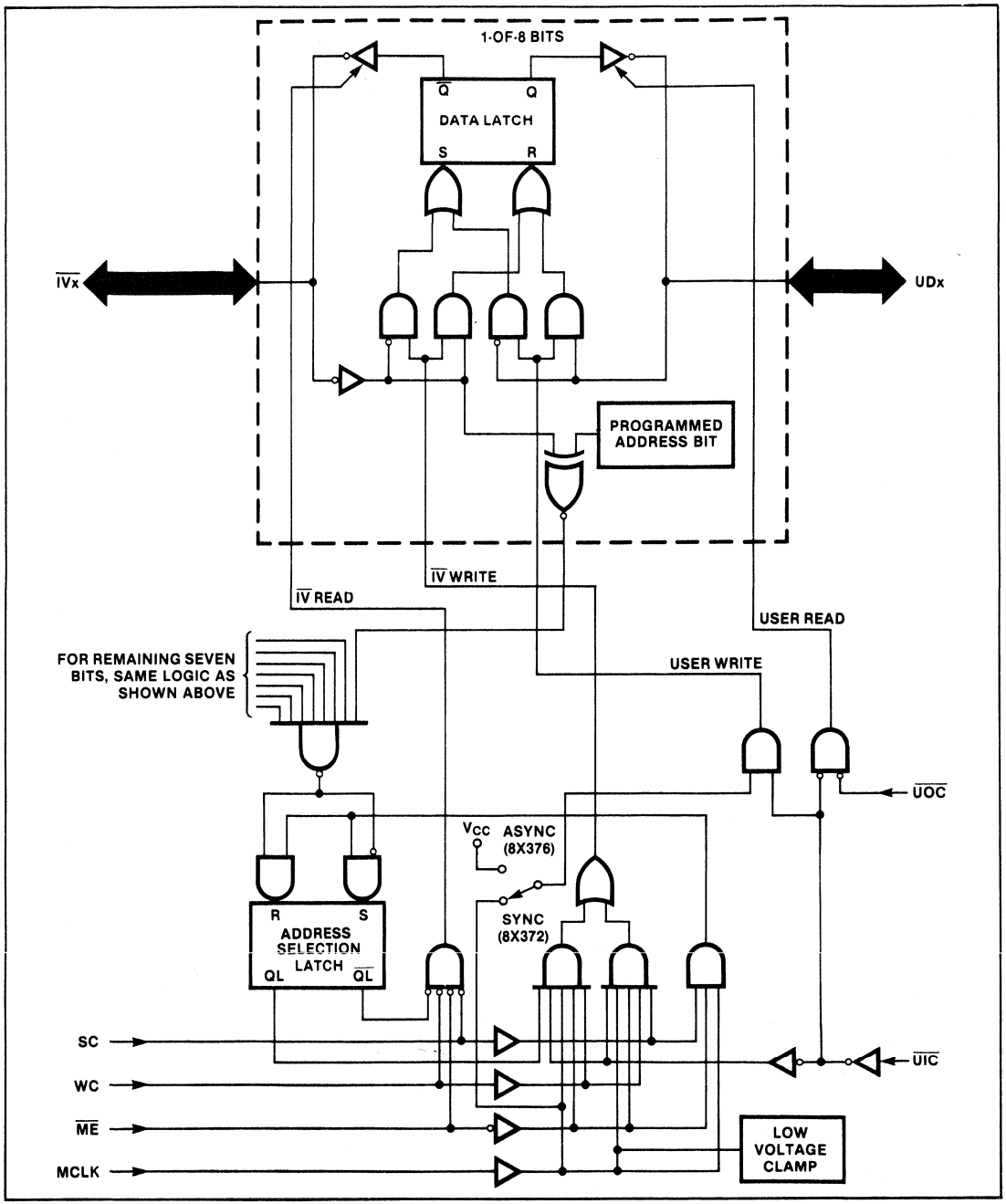


Figure 1. Logic Diagram for 8X372/8X376 I/O Ports

FUNCTIONAL OPERATION

UD Bus Control

As shown in Table 1, the User Data (UD) bus interface is controlled by the $\overline{UIC}$ and $\overline{UOC}$ inputs. For the 8X372, data input from the UD bus is written synchronously with MCLK, that is, with $\overline{UIC}$ low, information is written into the data latches only when MCLK is high. In the case of the 8X376, data input is asynchronous, in that, with $\overline{UIC}$ low, data is latched in without regard to the level of MCLK. Note. *To avoid the possibility of processor error when using the asynchronous 8X376, the $\overline{IV}$ bus should not be read during the time the data latches are changing due to user input.* Output drivers on the UD bus are enabled when UOC is low and $\overline{UIC}$ is high.

Table 1. INPUT/OUTPUT CONTROL OF UD BUS

$\overline{UIC}$	$\overline{UOC}$	MCLK	FUNCTION OF UD BUS	
			8X372	8X376
H	L	X	Output data	Output data
L	X	H	Input data	Input data
L	X	L	Inactive	Input data
H	H	X	Inactive	Inactive

X = don't care

$\overline{IV}$ Bus Control

Input/output control of the $\overline{IV}$ bus is shown in Table 2; this bus is controlled by SC, WC, $\overline{ME}$, MCLK and the current state of the internal address selection latch. AS shown in Table 2, $\overline{UIC}$ is required to indicate priority of the UD bus for data input operations. The selection latch in the I/O port stores the result of the most recent $\overline{IV}$ address selection. The latch is set when the internally preprogrammed address of the port matches the address on the $\overline{IV}$ bus during an address-selection operation (SC = MCLK = High/WC = Low). The latch is cleared when the two 8-bit address patterns are in disagreement. The $\overline{IV}$ bus can transfer data only when the selection latch is set. As shown in the APPLICATION DIAGRAM, the MicroController Left Bank ($\overline{LB}$) and Right Bank ($\overline{RB}$) outputs can control the $\overline{ME}$ inputs for two banks of I/O devices, thus, acting as a ninth address bit.

Table 2. INPUT/OUTPUT CONTROL OF IV BUS

$\overline{ME}$	SC	WC	MCLK	$\overline{UIC}$	SELECTION LATCH	FUNCTION OF $\overline{IV}$ BUS
L	L	L	X	X	Set	Output Data
L	L	H	H	H	Set	Input Data
L	H	L	H	X	X	Input Address*
L	H	H	H	H	X	Input data and address*
L	H	H	H	L	X	Input Address*
L	X	H	L	X	X	Inactive
L	H	X	L	X	X	Inactive
L	L	H	H	L	X	Inactive
L	L	X	X	X	Not Set	Inactive
H	X	X	X	X	X	Inactive

X = don't care

* Selection latch is updated

Data is written into the data latches of a selected device from the $\overline{IV}$ bus when WC, MCLK, and $\overline{UIC}$ are all high and $\overline{ME}$ is low. To prevent data-input conflicts, inputs from the $\overline{IV}$ bus are inhibited when $\overline{UIC}$ is low; under all other conditions, the $\overline{IV}$ and UD busses operate independently. Output drivers on the $\overline{IV}$ bus of a selected device are enabled when $\overline{ME}$, WC, and SC are all low and the address selection latch is set. With SC and WC both high (shaded entry of Table 2), the bit pattern present on $\overline{IV0}$-$\overline{IV7}$ is interpreted as both input data and $\overline{IV}$ address. Provided $\overline{UIC}$ is high, the data is latched into the data latches whether or not the I/O port has been previously selected. If the preprogrammed address of the I/O port matches the bit pattern on $\overline{IV0}$-$\overline{IV7}$ when SC and WC are both high, the selection latch is set; otherwise, it is reset. Note. *The MicroController never drives both SC and WC high at the same time.*

Bus Logic Levels

Data written into the I/O port from either bus will appear inverted when read from the other bus. Data written into either bus will not be inverted when read from the same bus. (Note. *A logic "1" in MicroController software corresponds to a high level on the UD bus even though the $\overline{IV}$ bus is inverted.* Both the 8X372 and 8X376 wakeup with the address selection latch in the unselected state and all data bits latched at the "logic 1" level (UD bus outputs high if enabled).

ADDRESS PROGRAMMING AND ADDRESS PROTECT

Programming Procedures

Both 8X372 and 8X376 can be programmed to respond to any address within a range of 0_{10} through 255_{10}. In an unprogrammed state, low level (≤0.8V) inputs on all $\overline{IV}$ bus lines (address 255_{10}) will select the device. To program a given address bit to match a high level (≥2.0V) input on the corresponding $\overline{IV}$ pin (a logical "0" to the MicroController), the counterpart UD-bus pin must be pulsed according to Table 3 and the following procedures:

Step 1: Set all control inputs to the inactive state—$\overline{UIC}$ = $\overline{UOC}$ = $\overline{ME}$ = V_{CC} and SC = WC = MCLK = GND; leave the UD and $\overline{IV}$ bus pins open.

Step 2: Increase V_{CC} to V_{CCP}.

Step 3: After V_{CC} has stabilized, apply a single programming pulse (Figure 2) to the user-bus bit that corresponds to the desired high-level $\overline{IV}$ address bit. The I/O port is programmed from the user bus (UD0-UD7) for addressing from the MicroController bus ($\overline{IV0}$-$\overline{IV7}$).

Step 4: Return V_{CC} to 0-volts. Note. *If the programming of all address bits is completed in less than 1-second, V_{CC} can remain at 9.0-volts for the required interval of time.*

Step 5: Step 1 through 3 are applicable to the programming of each address bit that requires a high-level $\overline{IV}$ match.

Table 3. PROGRAMMING SPECIFICATIONS

PARAMETERS	LIMITS			UNITS
	Min	Typ	Max	
V_{CCP} — Programming supply voltage:				
Address	8.75	9.0	9.25	V
Protect		0		V
Maximum time V_{CCP} >5.25V			1.0	Sec
Programming voltage:				
Address	8.75	9.0	9.25	V
Protect	8.75		9.25	V
Programming current:				
Address			5	*m*A
Protect			50	*m*A
t_r — Programming pulse/rise time:				
Address	10		100	μS
Protect	10		100	μS
t_ω — Programming pulse width	0.5		1.0	mS

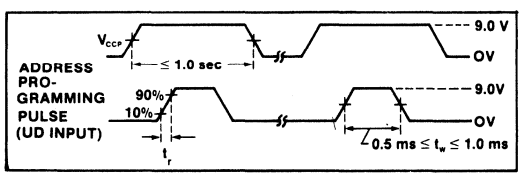


Figure 2. Address Programming Pulse

Step 6: To verify that the address is properly programmed, return V_{CC} to +5V, set $\overline{IV0}$-$\overline{IV7}$ to the desired (inverted) binary address pattern, set $\overline{ME}$ = WC = Low and SC = MCLK = High. If there are no programming errors, subsequent data written from $\overline{IV0}$-$\overline{IV7}$ (WC = High) will appear inverted on UD0-UD7.

Address Protect

After programming the I/O Port, steps should be taken to isolate the address circuits and make these circuits permanently immune to further change.

Step 1: Set V_{CC} and all control inputs to 0-volts (V_{CC} = $\overline{UIC}$ = $\overline{UOC}$ = $\overline{ME}$ = SC = WC = MCLK = 0V); $\overline{IV0}$-$\overline{IV7}$ = open circuit.

Step 2: Taking one pin at a time, apply a protect programming pulse (Figure 3) to each user-bus bit (UD0-UD7)—refer to Table 3 for min/max specifications pertaining to voltage and current.

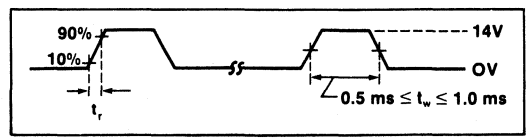


Figure 3. Protect Programming Pulse

Step 3: Verify that the address circuits for each bit is isolated by applying 9-volts, in turn, to each user-bus pin (UD0-UD7) and measuring less than 200 microamperes of input current. (Note. *Setup conditions are the same as those in Step 1.*)

DC ELECTRICAL CHARACTERISTICS

COMMERCIAL: 4.75V ≤ V_{CC} ≤ 5.25V, 0°C ≤ T_A ≤ 70°C
MILITARY: 4.5V ≤ V_{CC} ≤ 5.5V, -55°C ≤ T_C ≤ 125°C

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Power supply voltage[3]	+7	Vdc
V_{IN}	Input voltage[3]	+5.5	Vdc
T_{STG}	Storage temperature range	-65 to +150	°C

PARAMETER		TEST CONDITIONS	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	Supply Voltage		4.75	5	5.25	4.5	5	5.5	V
V_{IH}	High Level Input Voltage		2.0			2.0			V
V_{IL}	Low Level Input Voltage				0.8			0.8	V
V_{CL}	Input Clamp Voltage	V_{CC} = Min; I_I = -10mA			-1.5			-1.5	V
I_{IH}	High Level Input Current[1]	V_{CC} = Max; V_{IH} = 2.7V		5.0	100		5.0	100	μA
I_{IL}	Low Level Input Current[1]	V_{CC} = Max; V_{IL} = 0.5V		-350	-550		-350	-550	μA
V_{OL}	Low Level Output Voltage								
	$\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC} = Min; I_{OL} = 16mA			0.55		0.55	V	
	User Bus (UD0-UD7)	V_{CC} = Min; I_{OL} = 24mA			0.55			0.55	V
V_{OH}	High Level Output Voltage	V_{CC} = Min; I_{OH} = -3.2mA	2.4			2.4			V
I_{OS}	Short Circuit Output Current[2]								
	$\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC} = Max	-20			-20			mA
	UD Bus (UD0-UD7)	V_{CC} = Max	-10			-10			mA
I_{CC}	Supply Current	V_{CC} = Max; $\overline{ME}$ = $\overline{UOC}$ = V_{CC}		90	150		90	150	mA

NOTES:
1. The input current includes the Three-state leakage current of the output driver on the data lines.
2. Only one output may be shorted at a time.
3. These limits do not apply during address programming.

AC ELECTRICAL CHARACTERISTICS

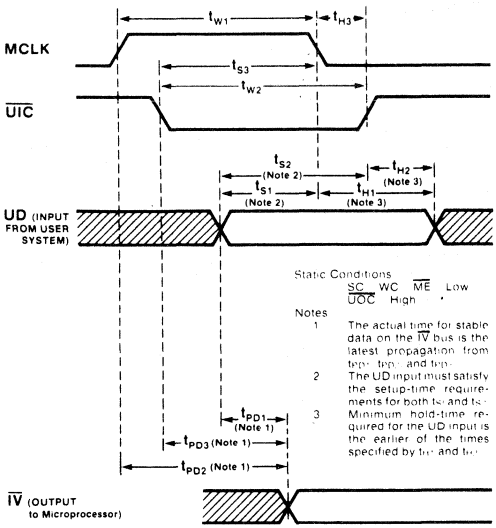
COMMERCIAL: 4.75V ≤ V_{CC} ≤ 5.25V, 0°C ≤ T_A ≤ 70°C
MILITARY: 4.5V ≤ V_{CC} ≤ 5.5V, -55°C = T_C ≤ 125°C
LOADING: See TEST LOADING CIRCUITS

PARAMETER		REFERENCES[1] FROM	REFERENCES[1] TO	TEST CONDITIONS	LIMITS (COMMERCIAL) Min	Typ	Max	LIMITS (MILITARY) Min	Typ	Max	UNIT
Pulse Widths:											
t_{W1}	Clock High	↑MCLK	↓MCLK		35			35			ns
t_{W2}	User Input Control	↓$\overline{UIC}$	↑$\overline{UIC}$	MCLK = High	35			35			ns
Propagation Delays:											
t_{PD1}	UD Propagation Delay	UD	$\overline{IV}$	MCLK = High SC = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			30			30	ns
t_{PD2}	UD Clock Delay (8X732 only)	↑MCLK	$\overline{IV}$	UD = Stable; SC = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			50			50	ns
t_{PD3}	UD Input Delay	↓$\overline{UIC}$	$\overline{IV}$	UD = Stable; MCLK = High; SC = WC = $\overline{ME}$ = Low			50			50	ns
t_{PD4}	$\overline{IV}$ Data Propagation Delay	$\overline{IV}$	UD	MCLK = WC = $\overline{UIC}$ = High; $\overline{ME}$ = $\overline{UOC}$ = SC = Low			45			45	ns
t_{PD5}	$\overline{IV}$ Data Clock Delay	↑MCLK	UD	WC = $\overline{UIC}$ = High; $\overline{IV}$ = Stable, $\overline{ME}$ = $\overline{UOC}$ = SC = Low			55			55	ns
Output Enable Timing:											
t_{OE1}	UD Output Enable	↓$\overline{UOC}$	UD	$\overline{UIC}$ = High			30			30	ns
t_{OE2}	UD Input Recovery	↑$\overline{UIC}$	UD	$\overline{UOC}$ = Low			30			30	ns
t_{OE3}	$\overline{IV}$ Data Master Enable	↓$\overline{ME}$	$\overline{IV}$	WC = SC = Low			22			25	ns
t_{OE5}	$\overline{IV}$ Data Write Recovery	↓WC	$\overline{IV}$	SC = $\overline{ME}$ = Low			25			25	ns
t_{OE6}	$\overline{IV}$ Data Select Recovery	↓SC	$\overline{IV}$	SC = $\overline{ME}$ = Low			25			25	ns
Output Disable Timing:											
t_{OD1}	UD Output Disable	↑$\overline{UOC}$	UD	$\overline{UIC}$ = High			25			25	ns
t_{OD2}	UD Input Override	↓$\overline{UIC}$	UD	$\overline{UOC}$ = Low			30			30	ns
t_{OD3}[2]	$\overline{IV}$ Data Master Disable	↑$\overline{ME}$	$\overline{IV}$	WC = SC = Low			20			20	ns
t_{OD4}[2]	$\overline{IV}$ Data Write Override	↑WC	$\overline{IV}$	SC = $\overline{ME}$ = Low			20			20	ns
t_{OD5}[2]	$\overline{IV}$ Data Select Override	↑SC	$\overline{IV}$	WC = $\overline{ME}$ = Low			20			20	ns
Setup Times:											
t_{S1}	UD Clock Setup Time (8X372 only)	UD	↓MCLK	$\overline{UIC}$ = Low	15			15			ns
t_{S2}	UD Control Setup Time	UD	↑$\overline{UIC}$	MCLK = High	15			15			ns
t_{S3}	User Input Control Setup Time (8X372 only)	↓$\overline{UIC}$	↓MCLK		25			25			ns
t_{S4}	$\overline{IV}$ Data Setup Time	$\overline{IV}$	↓MCLK	WC = High or SC = High; $\overline{ME}$ = Low; $\overline{UIC}$ = High	35			35			ns
t_{SS}[3]	$\overline{IV}$ Master Enable Setup Time	↓ME	↓MCLK	WC = High or SC = High; $\overline{UIC}$ = High	30			30			ns
t_{S6}	$\overline{IV}$ Write Control Setup Time	↑WC	↓MCLK	SC = $\overline{ME}$ = Low; $\overline{UIC}$ = High	30			30			ns

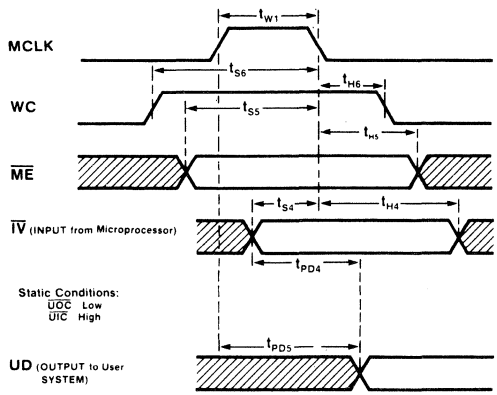
AC ELECTRICAL CHARACTERISTICS (Cont'd)

PARAMETER		REFERENCES FROM	REFERENCES TO	TEST CONDITIONS	LIMITS (COMMERCIAL) Min	Typ	Max	LIMITS (MILITARY) Min	Typ	Max	UNIT
t_{S7}	$\overline{IV}$ Select Control Setup Time	↑SC	↓MCLK	WC = $\overline{ME}$ = Low	30			30			ns
Hold Times:											
t_{H1}	UD Clock Hold Time (8X372 only)	↓MCLK	UD	$\overline{UIC}$ = Low	15			15			ns
t_{H2}	UD Control Hold Time	↑$\overline{UIC}$	UD	MCLK = High	15			15			ns
t_{H3}	User Input Control Hold Time (8X372 only)	↓MCLK	↑$\overline{UIC}$		0			0			ns
t_{H4}	$\overline{IV}$ Data Hold Time	↓MCLK	$\overline{IV}$	$\overline{WC}$ = High or SC = High; $\overline{ME}$ = Low, $\overline{UIC}$ = High	5			5			ns
t_{H5}[3]	$\overline{IV}$ Master Enable Hold Time	↓MCLK	↑$\overline{ME}$	$\overline{WC}$ = High or SC = High; $\overline{UIC}$ = High	0			0			ns
t_{H6}	$\overline{IV}$ Write Control Hold Time	↓MCLK	↓WC	SC = $\overline{ME}$ = Low; $\overline{UIC}$ = High	0			0			ns
t_{H7}	$\overline{IV}$ Select Control Hold Time	↓MCLK	↓SC	WC = $\overline{ME}$ = Low	0			0			ns

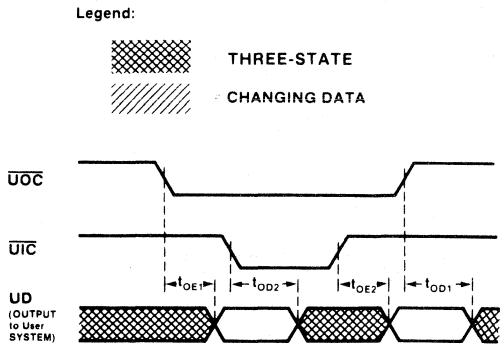
Notes:
1. All measurements to the $\overline{IV}$ bus assumes the address selection latch is set.
2. These parameters are measured with a capacitive loading of 50pf and represent the output driver turn-off time.
3. If $\overline{ME}$ is to be high (inactive), it must be setup before the rising edge and held after the falling edge of MCLK to avoid unintended writing into or selection of the I/O port.



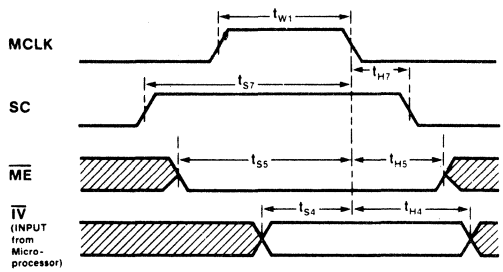
a. User Data Input Timing



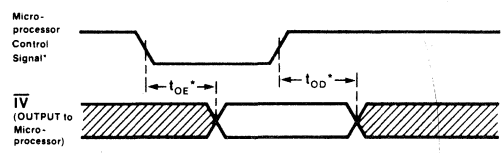
c. MicroController Write Cycle Timing



b. User Data Output Enable



d. MicroController Seclect Cycle Timing



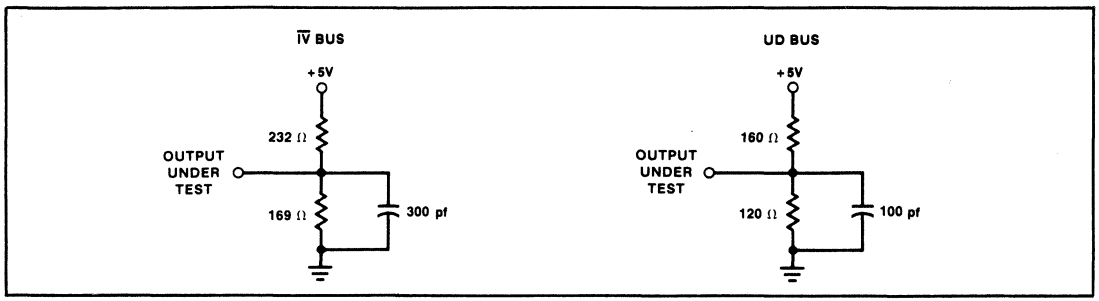
*PARAMETER KEY

MICROPROCESSOR CONTROL SIGNAL	AC TIMING PARAMETERS	STATIC CONDITIONS
ME	t_{OE3} t_{OD3}	SC WC LOW
WC	t_{OE5} t_{OD5}	SC ME LOW
SC	t_{OE6} t_{OD6}	WC ME LOW

e. MicroController Output Enable Timing

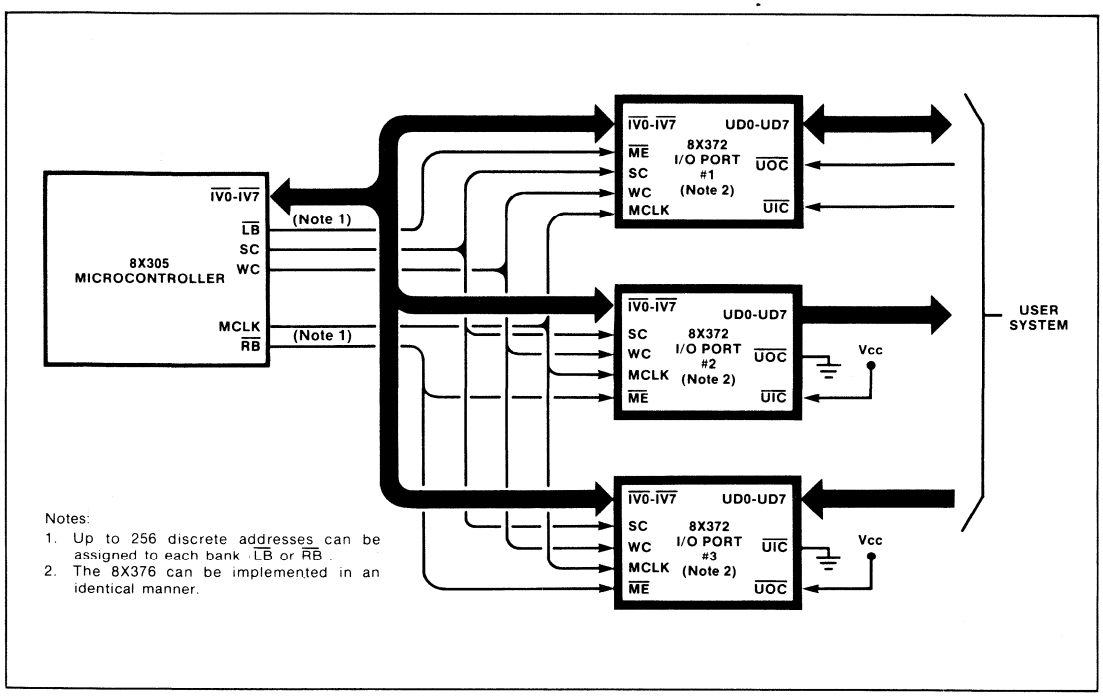
Figure 2. Timing Diagram

TEST LOADING CIRCUITS



APPLICATIONS

One way of using I/O Ports in a microprocessor-based system is shown in the following application diagram; there are many other ways of implementing I/O functions with these parts, both singly and in combination. By proper control of the $\overline{UIC}$ and $\overline{UOC}$ lines, the user can implement bidirectional data transfers, exercise system control, and/or read system status. In the concept shown here, I/O Port #1 is setup for bidirectional data transfers and I/O Ports #2 and #3, respectively, serve as dedicated output and input devices.



Notes:
1. Up to 256 discrete addresses can be assigned to each bank $\overline{LB}$ or $\overline{RB}$.
2. The 8X376 can be implemented in an identical manner.

ADDRESSABLE/BIDIRECTIONAL I/O PORT WITH PARITY

Originally published by Signetics January 1984

FEATURES

- **Two bidirectional 8-bit busses**
- **Independent bus operation (user bus priority for data entry)**
- **Parity generate/check logic with:**
  Odd/Even parity select
  Strobed error flag output
- **Synchronous data input**
- **Programmable MicroController port address**
- **Three-state TTL outputs (for all except parity error flag)**
- **High drive capabilities**
- **Power-up to predetermined state**
- **Directly compatible with 8X305 MicroController**
- **Single +5V supply**
- **0.6 inch, 28-pin DIP**

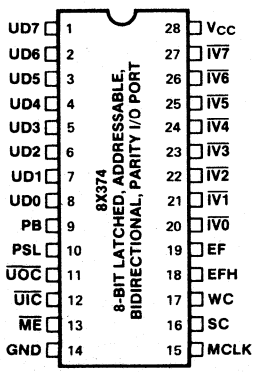
PRODUCT DESCRIPTION

The Signetics 8X374 is an addressable 8-bit I/O Port that features on-chip parity generate/check logic. The 8X374 port is designed for applications that require an 8-bit bidirectional interface element with parity-generate and parity-check capabilities. Typically, the 8X374 is used with the 8X305 MicroController and its associated Interface Vector (IV) bus.

As shown in the logic diagram of Figure 1, the 8X374 consists of eight identical latches, bits 0 through 7. These latches are accessed through either of two 8-bit busses, one connecting to the MicroController (IV bus) and the other to the user system (UD bus). Separate controls are provided for each bus and both busses operate independently, except when both attempt to input data at the same time. In such situations, the user bus always has priority. The data latches are transparent, in that, while either bus is enabled for input, all transitions in input data are propagated to the other bus, if enabled for output. The data latch in Figure 1 is common to both busses, that is, data traveling from the IV bus to the UD bus, or vice-versa, is latched and applied to the parity generate/check logic. The parity-bit latch is interfaced to the UD bus and latches the parity bit. The user can implement the parity features of the chip by simply selecting odd or even parity via the Parity SeLect (PSL) input pin. When data is output to the UD bus, a parity bit is generated and appended to each byte of data; for incoming data, parity is checked and the result is transmitted to an error-flag latch. The status of the latch (0 = no parity error/1 = parity error) is reflected by the Error Flag (EF) output pin. Operation of the error-flag latch is controlled by the Error Flag Hold (EFH) signal. With EFH low, the operation is transparent; when high the contents of the latch are frozen to avoid false errors while data latches are changing.

8X374 PACKAGE AND PIN DESIGNATIONS

N,F PACKAGE



TOP VIEW

ORDER NUMBERS

N8X374N, N8X374F
S8X374F/883B, S8X374F/883C

Pin No.	Identifier	Function
1-8	UD7-UD0	Three-state bidirectional User Data (UD) bus; UD0 corresponds to $\overline{IV0}$.
9	PB	User port Parity Bit I/O pin.
10	PSL	Parity SeLect input control; even parity = 1 and odd parity = 0.
11	$\overline{UOC}$	User Output Control — active low input to enable data output from UD0-UD7.
12	$\overline{UIC}$	User Input Control — active low input to enable data input from UD0-UD7.
13	$\overline{ME}$	Master Enable — active low input to enable the IV bus for data input, data output, or IV address selection/deselection; UD-bus operations are unaffected.
14	GND	Ground
15	MCLK	Master Clock — active high input from MicroController used to strobe data into the data latches; MCLK also synchronizes IV address selection.
16	SC	Select Command — active high input from MicroController to enable IV address input from the IV bus for device selection.
17	WC	Write Command — active high input from MicroController to enable the writing of data into the data latches from the IV bus, provided $\overline{UIC}$ is not low.
18	EFH	Error Flag Hold signal to control error-flag latch. When low, latch operation is transparent; when high, contents of latch are frozen.
19	EF	Error Flag output; no parity error = 0 parity error = 1.
20-27	$\overline{IV0}$-$\overline{IV7}$	Interface Vector (Input/Output Bus), three-state, bidirectional, MicroController data bus; $\overline{IV0}$ corresponds to UD0.
28	VCC	+5V power supply.

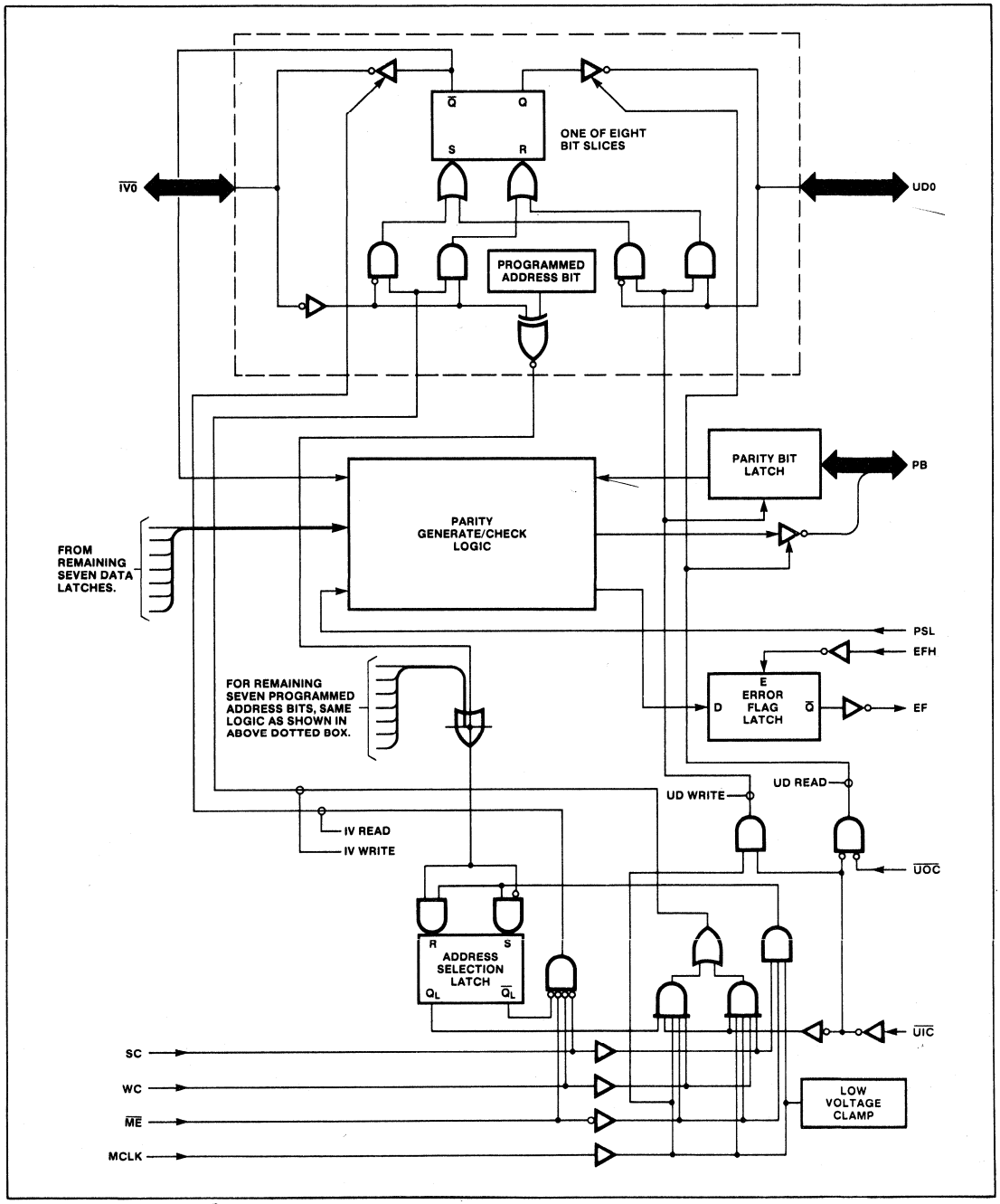


Figure 1. Logic Diagram for 8X374 I/O Port

The 8X374 is available with either preprogrammed addresses (0_{10} to 255_{10}) or unprogrammed; the device can be field-programmed over the same address range as the preprogrammed port. Input/Output operations to the MicroController bus can begin once the 8X374 enabling address has been selected and appropriate control signals from the IV bus are generated. Port selection is implemented by putting the 8X374 address (0_{10} to 255_{10}) on the IV bus. Once selected, the I/O port remains selected until a different port address is put on the bus.

With appropriate control inputs, data is accessible on the UD bus at all times. A Master Enable (ME) input, which is typically connected to the Left Bank (LB) or Right Bank (RB) output of the MicroController, provides the capability of organizing the IV bus into two separate and independent banks of I/O devices.

FUNCTIONAL OPERATION

UD Bus Control

As shown in Table 1, the User Data (UD) bus and parity-bit interface are controlled by the UIC and UOC inputs. Data from the UD bus is written synchronously with MCLK, that is with UIC low, information is written into the data latches only when MCLK is high. Output drivers on the UD bus are enabled when UOC is low and UIC is high.

Table 1. INPUT/OUTPUT CONTROL OF UD BUS

			Function of UD Bus	
UIC	UOC	MCLK	8-Bit Data Bus	Parity Bit
H	L	X	Output data	Output parity
L	X	H	Input data	Input parity
L	X	L	Inactive	Inactive
H	H	X	Inactive	Inactive

X = Don't Care.

IV Bus Control

Input/Output control of the IV bus is shown in Table 2; this bus is controlled by SC, WC, ME, MCLK and the current state of the internal address selection latch. As shown in Table 2, UIC is required to indicate priority of the UD bus for data input operations. The selection latch in the I/O port stores the result of the most recent IV address selection. The latch is set when the internally preprogrammed address of the port matches the address on the IV bus during an address-selection operation (SC = MCLK = High; ME = WC = Low). The latch is cleared when the two 8-bit address patterns are in disagreement. The IV bus can transfer data only when the selection latch is set. As shown in the APPLICATION DIAGRAM, the 8X305 Left Bank (LB) and Right Bank (RB) outputs can control the ME inputs for two banks of I/O devices, thus, acting as a ninth address bit.

Table 2. INPUT/OUTPUT CONTROL OF IV BUS

ME	SC	WC	MCLK	UIC	Selection Latch	Function of IV Bus
L	L	L	X	X	Set	Output Data
L	L	H	H	H	Set	Input Data
L	H	L	H	X	X	Input Address*
L	X	H	L	X	X	Inactive
L	H	X	L	X	X	Inactive
L	L	H	H	L	X	Inactive
L	L	X	X	X	Not Set	Inactive
H	X	X	X	X	X	Inactive

X = Don't Care. *Selection latch is updated.

Data is written into the data latches of a selected device from the IV bus when WC, MCLK and UIC are all high and ME is low. To prevent data-input conflicts, inputs from the IV bus are inhibited when UIC is low; under all other conditions, the IV and UD busses operate independently. Output drivers on the IV bus of a selected device are enabled when ME, WC, and SC are all low and the address selection latch is set.

Parity Generate/Check Logic

The Parity Bit (PB) pin provides both parity-generate and parity-check capabilities according to user data bus controls. With UIC low (active), a parity check is performed on the input data stream; with UOC low (active) and UIC high, the 8X374 generates the parity-bit for the output, data stream. The user can select odd or even parity via the Parity SeLect (PSL) input control, 1 = even parity and 0 = odd parity. As data and parity are input to the data latches and the parity-bit latch from the UD bus and PB line (Figure 1), parity errors (if any) are continuously detected by the parity-check logic. Parity error status enters the error flag latch (if enabled) and appears at the EF output pin. The error latch can be strobed by the Error Flag Hold (EFH) control to latch in valid error status; otherwise, the error flag is transparent to the user. (Note: If the sytem uses less than eight data bits, keeping zeros in unused data latches preserves proper parity operation.)

Bus Logic Levels

Data written into the I/O port from either bus will appear inverted when read from the other bus. Data written into either bus will not be inverted when read from the same bus. (Note: A logic "1" in MicroController software corresponds to a high level on the UD bus even though the IV bus

is inverted.) The 8X374 wakes up with the address selection latch in the unselected state, all data bits latched at the "logic 1" level (UD bus outputs high if enabled), and the EF output high.

ADDRESS PROGRAMMING AND ADDRESS PROTECT

Programming Procedures

The 8X374 can be programmed to respond to any address within a range of 0_{10} through 255_{10}. In an unprogrammed state, low level (≤ 0.8 V) inputs on all IV bus lines (address 255_{10}) will select the device. To program a given address bit to match a high level (≥ 2.0 V) input on the corresponding IV pin (a logical "0" to the MicroController), the counterpart UD-bus pin must be pulsed according to Table 3 and the following procedures:

Step 1: Set all control inputs to the inactive state, UIC = UOC = ME = V_{CC} and SC = WC = MCLK = 0 V; leave the UD and IV bus pins open.

Table 3. PROGRAMMING SPECIFICATIONS

Parameters	Limits			Units
	Min.	Typ.	Max.	
V_{CCP} — Programming supply voltage:				
Address	8.75	9.0	9.25	V
Protect		0		V
Maximum Time V_{CC} > 5.25 V			1.0	sec
Programming voltage:				
Address	8.75	9.0	9.25	V
Protect	8.75		9.25	V
Programming current:				
Address			5	mA
Protect			50	mA
t_r — Programming pulse rise time:				
Address	10		100	μs
Protect	10		100	μs
t_W — Programming pulse width	0.5		1.0	ms

Step 2: Increase V_{CC} to V_{CCP}.

Step 3: After V_{CC} has stabilized, apply a single programming pulse (Figure 2) to the user-bus bit that corresponds to the desired high-level IV address bit. The I/O port is programmed from the user bus (UD0-UD7) for addressing from the MicroController bus (IV0-IV7).

Step 4: Return V_{CC} to 0 volts. (Note: If the programming of all address bits is completed in less than one second, V_{CC} can remain at V_{CCP} for the required interval of time.)

Step 5: Step 1 through Step 3 are applicable to the programming of each address bit that requires a high-level IV match.

Step 6: To verify that the address is properly programmed, return V_{CC} to +5 V and set IV0-IV7 to the desired address pattern (inverted). Set ME = WC = Low and SC = MCLK = High to select the programmed I/O port. With ME = SC = Low and WC = MCLK = High, write an 8-bit pattern to the port. If there are no programming errors, the transmitted data pattern will appear inverted at UD0-UD7 of selected port.

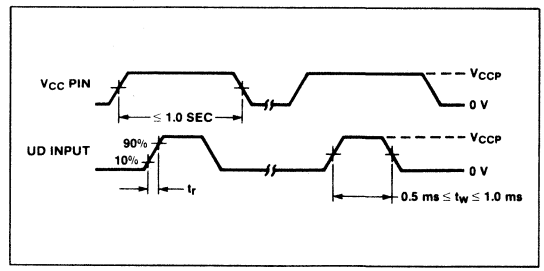


Figure 2. Address Programming Pulse

ADDRESS PROTECT

After programming the I/O Port, optional steps can be taken to isolate the fuse circuits and to make these circuits permanently immune to further change.

Step 1: Set V_{CC} and all control inputs to 0 volts; V_{CC} = UIC = UOC = ME = SC = WC = MCLK = 0V, IV0-IV7 = open circuit.

Step 2: Taking one pin at a time, apply a protect programming pulse (Figure 3) to each user-bus bit (UD0-UD7). Refer to Table 3 for min/max specifications pertaining to voltage and current.

Step 3: Verify that the address circuits for each bit are isolated by applying V_{CCP}, in turn, to each user-bus pin (UD0-UD7) and measuring less than 200 microamperes of input current. (Note: Setup conditions are the same as those in Step 1.)

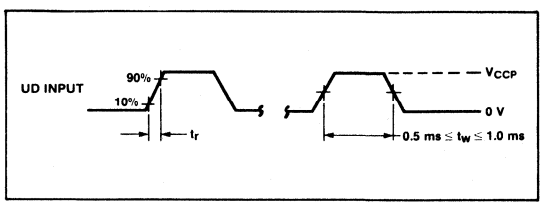


Figure 3. Protect Programming Pulse

ABSOLUTE MAXIMUM RATINGS

	Parameter	Rating	Unit
V_{CC}	Power supply voltage[3]	+7	V DC
V_{IN}	Input voltage[3]	+5.5	V DC
T_{STG}	Storage temperature range	-65 to +150	°C

DC ELECTRICAL CHARACTERISTICS

COMMERCIAL: V_{CC} = 5 V (±5%); $T_A \geq 0°C$
$T_A \leq 70°C$
MILITARY: V_{CC} = 5 V (±10%); $T_A \geq -55°C$
$T_C \leq 125°C$

Parameter		Test Conditions	Limits (Commercial)			Limits (Military)			Unit
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	Supply Voltage		4.75	5	5.25	4.5	5	5.5	V
V_{IH}	High Level Input Voltage		2.0			2.0			V
V_{IL}	Low Level Input Voltage				0.8			0.8	V
V_{CL}	Input Clamp Voltage	V_{CC} = Min; I_I = -10 mA			-1.5			-1.5	V
I_{IH}	High Level Input Current[1]	V_{CC} = Max; V_{IH} = 2.7 V		5.0	100		5.0	250	μA
I_{IL}	Low Level Input Current[1]	V_{CC} = Max; V_{IL} = 0.5 V		-350	-550		-350	-550	μA
V_{OL}	Low Level Output Voltage $\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC} = Min; I_{OL} = 16 mA			0.55			0.55	V
	User Bus (UD0-UD7) and PB	V_{CC} = Min; I_{OL} = 24 mA			0.55			0.55	V
	EF	V_{CC} = Min; I_{OL} = 8 mA			0.55			0.55	V
V_{OH}	High Level Output Voltage EF	V_{CC} = Min; I_{OH} = -1 mA	2.4			2.4			V
	Others	V_{CC} = Min; I_{OH} = -3.2 mA	2.4			2.4			V
I_{OS}	Short Circuit Output Current[2] $\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC} = Max	-20			-20			mA
	UD Bus (UD0-UD7)	V_{CC} = Max	-10			-10			mA
I_{CC}	Supply Current	V_{CC} = Max; $\overline{ME}$ = $\overline{UOC}$ = V_{CC}		90	150		90	160	mA

Notes:
1. The input current includes the high-Z leakage current of the output drivers ($\overline{IV0}$-$\overline{IV7}$, UD0-UD7) on the data lines.
2. Only one output may be shorted at a time for testing purposes.
3. These limits do not apply during address programming.

AC ELECTRICAL CHARACTERISTICS

COMMERCIAL: V_{CC} = 5 V (±5%); $T_A \geq 0°C$, $T_A \leq 70°C$
MILITARY: V_{CC} = 5 V (±10%); $T_A \geq -55°C$, $T_C \leq 125°C$
LOADING: See TEST LOADING CIRCUITS

Parameter		References From	References To	Test Conditions[1]	Limits (Commercial) Min	Typ	Max	Limits (Military) Min	Typ	Max	Unit
Pulse Widths:											
t_{W1}	Clock High	↓MCLK	↑MCLK		35			35			ns
t_{W2}	User Input Control	↓$\overline{UIC}$	↑$\overline{UIC}$	MCLK = High	35			35			ns
Propagation Delays:											
t_{PD1}	UD Propagation Delay	UD	$\overline{IV}$	MCLK = High; SC = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			40			40	ns
t_{PD2}	UD Clock Delay	↑MCLK	$\overline{IV}$	UD = Stable; SC = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			50			50	ns
t_{PD3}	UD Input Delay	↓$\overline{UIC}$	$\overline{IV}$	UD = Stable; MCLK = High; SC = WC = $\overline{ME}$ = Low			50			50	ns
t_{PD4}	$\overline{IV}$ Data Propagation Delay	$\overline{IV}$	UD	MCLK = WC = $\overline{UIC}$ = High; $\overline{ME}$ = $\overline{UOC}$ = SC = Low			45			45	ns
t_{PD5}	$\overline{IV}$ Data Clock Delay	↑MCLK	UD	WC = $\overline{UIC}$ = High; $\overline{IV}$ = Stable, $\overline{ME}$ = $\overline{UOC}$ = SC = Low			55			55	ns
t_{PD6}	Error Flag Propagation Delay	UD, PB	EF	MCLK = High; $\overline{UIC}$ = EFH = Low			55			55	ns
t_{PD7}	Parity Generate Propagation Delay	$\overline{IV}$	PB	MCLK = WC = $\overline{UIC}$ = High; $\overline{UOC}$ = $\overline{ME}$ = Low			55			55	ns
t_{PD8}	Error Flag Strobe Delay[3]	↓EFH	EF				20			20	ns
Output Enable Timing:											
t_{OE1}	UD Output Enable	↓$\overline{UOC}$	UD, PB	$\overline{UIC}$ = High			30			30	ns
t_{OE2}	UD Input Recovery	↓$\overline{UIC}$	UD, PB	$\overline{UOC}$ = Low			30			30	ns
t_{OE3}	$\overline{IV}$ Data Master Enable	↓$\overline{ME}$	$\overline{IV}$	WC = SC = Low			22			25	ns
t_{OE4}	$\overline{IV}$ Data Write Recovery	↓WC	$\overline{IV}$	SC = $\overline{ME}$ = Low			25			25	ns
t_{OE5}	$\overline{IV}$ Data Select Recovery	↓SC	$\overline{IV}$	SC = $\overline{ME}$ = Low			25			25	ns

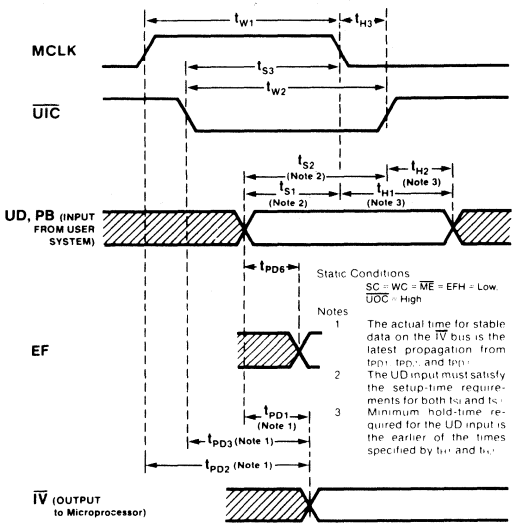
AC ELECTRICAL CHARACTERISTICS (Continued)

Parameter	References		Test Conditions[1]	Limits (Commercial)			Limits (Military)			Unit
	From	To		Min	Typ	Max	Min	Typ	Max	
Output Disable Timing:										
t_{OD1} UD Output Disable	↓$\overline{UOC}$	UD, PB	$\overline{UIC}$ = High			25			25	ns
t_{OD2} UD Input Override	↓$\overline{UIC}$	UD, PB	$\overline{UOC}$ = Low			30			30	ns
t_{OD3} $\overline{IV}$ Data Master Disable	↓$\overline{ME}$	$\overline{IV}$	WC = SC = Low			20			20	ns
t_{OD4} $\overline{IV}$ Data Write Override	↓WC	$\overline{IV}$	SC = $\overline{ME}$ = Low			20			20	ns
t_{OD5} $\overline{IV}$ Data Select Override	↓SC	$\overline{IV}$	WC = $\overline{ME}$ = Low			20			20	ns
Setup Times:										
t_{S1} UD Clock Setup Time	UD, PB	↑MCLK	$\overline{UIC}$ = Low	15			15			ns
t_{S2} UD Control Setup Time	UD, PB	↓$\overline{UIC}$	MCLK = High	15			15			ns
t_{S3} User Input Control Setup Time	↓$\overline{UIC}$	↑MCLK		25			25			ns
t_{S4} $\overline{IV}$ Data Setup Time	$\overline{IV}$	↑MCLK	WC = High or SC = High; $\overline{ME}$ = Low; $\overline{UIC}$ = High	35			35			ns
t_{S5}[2] $\overline{IV}$ Master Enable Setup Time	↓$\overline{ME}$	↑MCLK	WC = High or SC = High; $\overline{UIC}$ = High	30			30			ns
t_{S6} $\overline{IV}$ Write Control Setup Time	↓WC	↑MCLK	SC = $\overline{ME}$ = Low; $\overline{UIC}$ = High	30			30			ns
t_{S7} $\overline{IV}$ Select Control Setup Time	↓SC	↑MCLK	WC = $\overline{ME}$ = Low	30			30			ns
Hold Times:										
t_{H1} UD Clock Hold Time	↑MCLK	UD, PB	$\overline{UIC}$ = Low	15			15			ns
t_{H2} UD Control Hold Time	↓$\overline{UIC}$	UD, PB	MCLK = High	15			15			ns
t_{H3} User Input Control Hold Time	↑MCLK	↓$\overline{UIC}$		0			0			ns
t_{H4} $\overline{IV}$ Data Hold Time	↑MCLK	$\overline{IV}$	WC = High or SC = High; $\overline{ME}$ = Low, $\overline{UIC}$ = High	5			5			ns
t_{H5}[2] $\overline{IV}$ Master Enable Hold Time	↑MCLK	↑$\overline{ME}$	WC = High or SC = High; $\overline{UIC}$ = High	0			0			ns
t_{H6} $\overline{IV}$ Write Control Hold Time	↑MCLK	↑WC	SC = $\overline{ME}$ = Low; $\overline{UIC}$ = High	0			0			ns
t_{H7} $\overline{IV}$ Select Control Hold Time	↑MCLK	↓SC	WC = $\overline{ME}$ = Low	0			0			ns

Notes:
1. All measurements to the $\overline{IV}$ bus assumes the address selection latch is set.
2. If $\overline{ME}$ is to be high (inactive), it must be setup before the rising edge and held after the falling edge of MCLK to avoid unintended writing into or selection of the I/O port.
3. Parameters are measured by holding $\overline{UIC}$ = High and MCLK = Low and changing the state of the PSL input before each EFH pulse.

TIMING DIAGRAMS

USER DATA INPUT TIMING

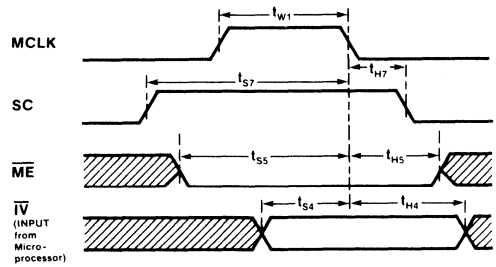


Legend:

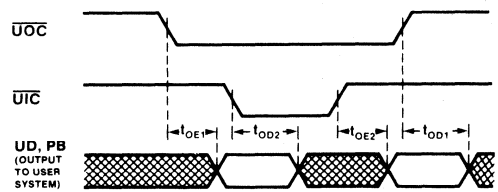
= HIGH-Z STATE

= CHANGING DATA

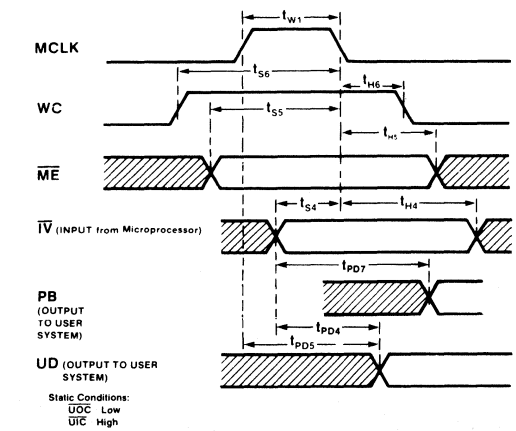
MICROCONTROLLER SELECT CYCLE TIMING



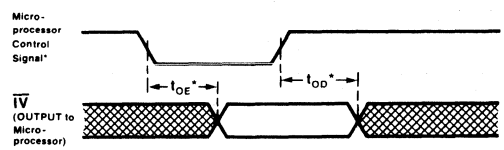
USER DATA OUTPUT ENABLE TIMING



MICROCONTROLLER WRITE CYCLE TIMING



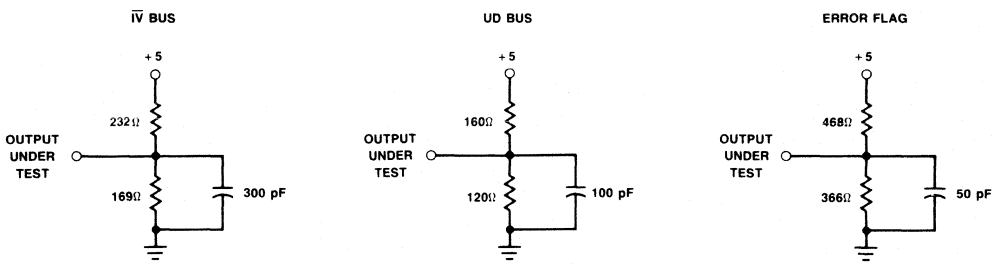
MICROCONTROLLER OUTPUT ENABLE TIMING



***PARAMETER KEY**

MICROPROCESSOR CONTROL SIGNAL	AC TIMING PARAMETERS	STATIC CONDITIONS
$\overline{ME}$	t_{OE3} t_{OD3}	SC = WC = LOW
WC	t_{OE5} t_{OD5}	SC = $\overline{ME}$ = LOW
SC	t_{OE6} t_{OD6}	WC = $\overline{ME}$ = LOW

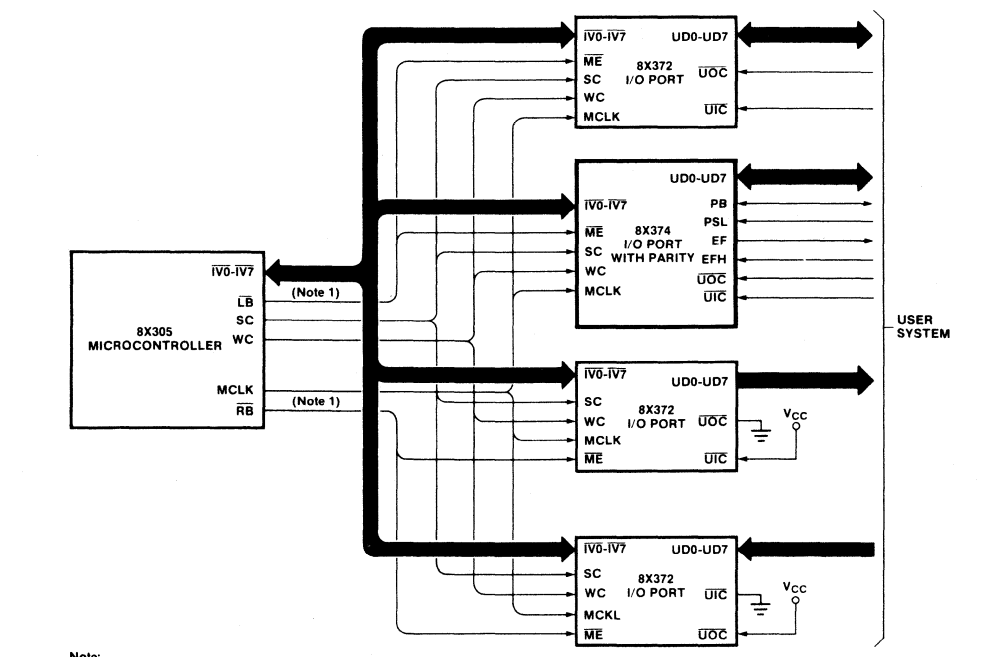
TEST LOADING CIRCUITS



APPLICATIONS

As shown in the following diagram, the 8X374 can be used with other I/O ports to provide a complete range of input/output functions. By proper control of the UIC and UOC lines, the user can perform bidirectional data transfers, exercise system control, read system status and, by using the 8X374, implement a bidirectional parity-controlled data stream. To use the parity capabilities, the user need only select even or odd parity (PSL = 1 or 0) and connect the PB pin to the system parity bit. The EFH and EF pins can be wired according to system requirements.

APPLICATIONS DIAGRAM



Note:
1. Up to 256 discrete addresses can be assigned to each bank ($\overline{LB}$ or $\overline{RB}$).

4-INPUT/4-OUTPUT ADDRESSABLE I/O PORT

Originally published by Signetics January 1984

FEATURES

- **Bidirectional 8-bit MicroController ($\overline{IV}$) bus**
- **User bus—four input bits and four output bits**
- **Independent bus operation**
- **Synchronous user data input**
- **Programmed MicroController port address**
- **Three-state TTL outputs with high-drive capabilities**
- **Power-up to predetermined logic state**
- **Directly compatible with 8X305 or 8X300 MicroControllers**
- **Single +5V supply**
- **0.4 inch 24-pin DIP**

PRODUCT DESCRIPTION

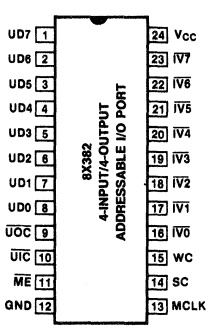
The 8X382 I/O Port is an addressable, three-state device designed for use as an interface element in systems that use TTL-compatible busses. Typically, the 8X382 is used with the 8X305 MicroController and its associated Interface Vector ($\overline{IV}$) bus; however, it can also be used with the 8X300 MicroController or an equivalent microprocessor. The 8X382 is functionally the same and pin-for-pin compatible with the older 8X42; however, the new port features better performance, increased drive current, and improved programming procedures.

As shown in the logic diagram of Figure 1, the I/O port consists of eight data latches—bits 0 through 7. These latches are accessed through either of two busses—an 8-bit bidirectional $\overline{IV}$ bus connected to the MicroController and a User Data (UD) bus consisting of four dedicated inputs (bits UD0 through UD3) and four dedicated outputs (bits UD4 through UD7). All eight bits may be read from or four data bits ($\overline{IV4}$-$\overline{IV7}$) can be written into via the $\overline{IV}$ bus; eight bits of I/O address can be written from the $\overline{IV}$ bus. Separate controls are provided for each bus and both busses operate independently. The I/O data latches are transparent, in that, when either bus is enabled for input, all transitions in input data are propagated to the other bus, if that bus in enabled for output.

The 8X382 is available with preprogrammed addresses (0_{10} through 255_{10}); it can also be field-programmed over the same address range. Input/output operations can begin once the I/O port is selected and appropriate control signals are generated. Port selection is implemented by putting the I/O port address (0_{10}-255_{10}) on the $\overline{IV}$ bus; once selected, the I/O port remains selected until a different "port address" is put on the bus. Thus, software overhead is minimized. Data is accessible on the UD bus at all times. A Master Enable ($\overline{ME}$) input, which is typically connected to the Left Bank ($\overline{LB}$) or Right Bank ($\overline{RB}$) output of the MicroController, provides the capability of organizing the $\overline{IV}$ bus into two separate and independent banks of I/O devices.

8X382 PACKAGE and PIN DESIGNATIONS

N, I PACKAGE



TOP VIEW

ORDER NUMBERS

N8X382N, N8X382I
S8X382I/883B, 8X382I/883C

PIN NO.	IDENTIFIER	FUNCTION
1-4	UD7-UD4	Three-state, dedicated output lines for user data; UD7 corresponds to $\overline{IV7}$.
5-8	UD3-UD0	Dedicated input lines for user data; UD0 corresponds to $\overline{IV0}$.
9	$\overline{UOC}$	User Output Control—active low input to enable data output to UD4-UD7.
10	$\overline{UIC}$	User Input Control—active low input to enable data input to UD0-UD3.
11	$\overline{ME}$	Master Enable—active low input to enable the $\overline{IV}$ bus for data input, data output, or $\overline{IV}$ address selection/deselection; UD-bus operations are unaffected.
12	GND	Ground.
13	MCLK	Master Clock—active high input (from MicroController) used to strobe data into data latches from the $\overline{IV}$ bus and bits UD0-UD3 of the UD bus; MCLK also synchronizes $\overline{IV}$ address selection.
14	SC	Select Command—active high input (from MicroController) to enable $\overline{IV}$ address input from the $\overline{IV}$ bus for device selection.
15	WC	Write Command—active high (from MicroController) to enable the writing of data into the data latches from the $\overline{IV}$ bus, provided $\overline{UIC}$ is not low.
16-23	$\overline{IV0}$-$\overline{IV7}$	Interface Vector (Input/Output Bus)—three-state, bidirectional, data bus; $\overline{IV0}$ corresponds to UD0.
24	V_{CC}	Supply Voltage.

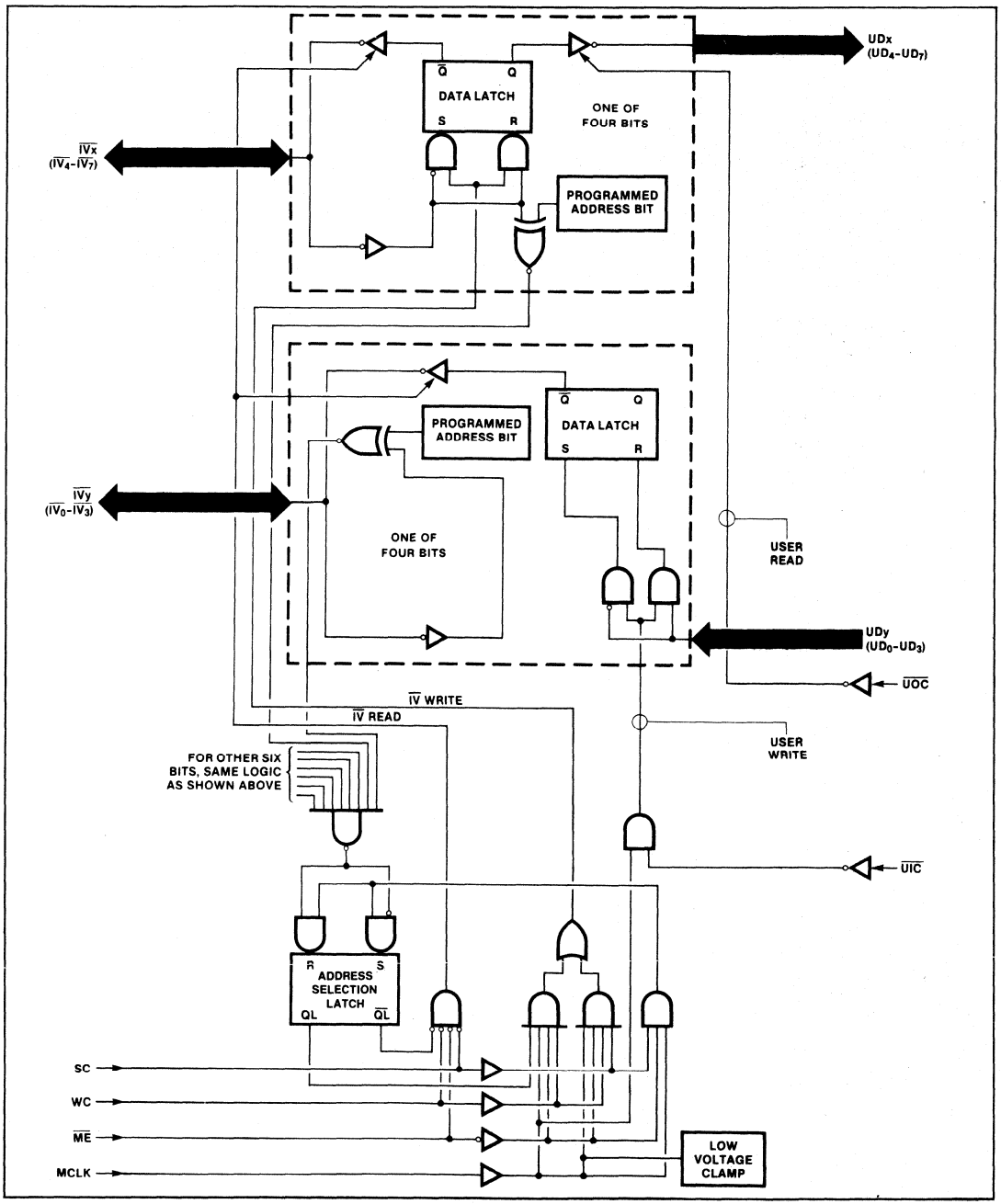


Figure 1. Logic Diagram for 8X382 I/O Port

FUNCTIONAL OPERATION

UD Bus Control

As shown in Table 1, the User Data-bus interface is controlled by the $\overline{UIC}$ and $\overline{UOC}$ inputs. Data input to UD0-UD3 is synchronous with MCLK, that is, with $\overline{UIC}$ low, information is written into the data latches only when MCLK is high. The output drivers of UD4-UD7 bus are enabled when $\overline{UOC}$ is low

Table 1. INPUT/OUTPUT CONTROL OF UD BUS

$\overline{UIC}$	$\overline{UOC}$	MCLK	FUNCTION OF UD BUS	
			UD0-UD3	UD4-UD7
H	L	X	Inactive	Output Data
L	X	H	Input Data	Inactive
L	X	L	Inactive	Inactive
H	H	X	Inactive	Inactive

X = don't care

$\overline{IV}$ Bus Control

Input/output control of the $\overline{IV}$ bus is shown in Table 2; this bus is controlled by SC, WC, $\overline{ME}$, MCLK and the current state of an internal address selection latch. The address selection latch in the I/O port stores the result of the most recent $\overline{IV}$ address selection. The latch is set when the internally preprogrammed address of the port matches the address on the $\overline{IV}$ bus during an address-selection operation (SC=MCLK=High/WC=Low). The latch is cleared when the two 8-bit address patterns are in disagreement. The $\overline{IV}$ bus can transfer data only when the selection latch is set. The MicroController Left Bank ($\overline{LB}$) and Right Bank ($\overline{RB}$) outputs can control the $\overline{ME}$ inputs for two banks of I/O devices, thus, acting as ninth address bit.

Data is written into the data latches of a selected device from the $\overline{IV}$ bus when WC = MCLK = High and $\overline{ME}$ = Low. Output drivers on the $\overline{IV}$ bus of the device with the address latch set are enabled with $\overline{ME}$, WC, and SC low. With SC and WC both high (shaded entry of Table 2), the bit pattern present on $\overline{IV0}$-$\overline{IV7}$ is interpreted as both input data ($\overline{IV4}$-$\overline{IV7}$ only) and $\overline{IV}$ address. The data in $\overline{IV4}$-$\overline{IV7}$ is latched in whether or not the I/O port has been previously selected. If the preprogrammed address of the I/O port matches the bit pattern on $\overline{IV0}$-$\overline{IV7}$ when SC and WC are both high, the selection latch is set; otherwise, it is reset. (Note. *The MicroController never drives both SC and WC high at the same time.*)

Table 2. INPUT/OUTPUT CONTROL OF IV BUS

$\overline{ME}$	SC	WC	MCLK	SEL. LATCH	FUNCTION OF $\overline{IV}$ BUS
L	L	L	X	Set	Output Data
L	L	H	H	Set	Input Data ($\overline{IV4}$-$\overline{IV7}$ only)
L	H	L	H	X	Input Address*
L	H	H	H	X	Input Data ($\overline{IV4}$-$\overline{IV7}$ only) and address*
L	X	H	L	X	Inactive
L	H	X	L	X	Inactive
L	L	X	X	Not set	Inactive
H	X	X	X	X	Inactive

X don't care

* Selection latch is updated

Bus Logic Levels

Data written into the I/O port from either bus will appear inverted when read from the other bus. Data written into either bus will not be inverted when read from the same bus. (Note. *A logic "1" in MicroController software corresponds to a high level on the UD bus even though the IV bus is inverted).* The 8X382 wakes up in the unselected state with all data bits latched at the "logic 1" level (UD bus outputs high if enabled).

ADDRESS PROGRAMMING AND ADDRESS PROTECT

Programming Procedures

The 8X382 can be programmed to respond to any address within a range of 0_{10} through 255_{10}. In an unprogrammed state, low level (≤0.8V) inputs on all $\overline{IV}$ bus lines (address 255_{10}) will select the device. To program a given address bit to match a high level (≥2.0V) input on the corresponding $\overline{IV}$ pin (a logical "0" to the MicroController), the counterpart UD-bus pin must be pulsed according to Table 3 and the following procedures:

Step 1: Set all control inputs to the inactive state—$\overline{UIC}$ = $\overline{UOC}$ = $\overline{ME}$ = V_{CC} and SC = WC = MCLK = GND; leave the UD and $\overline{IV}$ bus pins open.

Step 2: Increase V_{CC} to V_{CCP}.

Step 3: After V_{CC} has stabilized, apply a single programming pulse (Figure 2) to the user-bus bit that corresponds to the desired high-level $\overline{IV}$ address bit. The I/O port is programmed from the user bus (UD0-UD7) for addressing from the MicroController bus ($\overline{IV0}$-$\overline{IV7}$).

Table 3. PROGRAMMING SPECIFICATIONS

PARAMETERS	LIMITS			UNITS
	Min	Typ	Max	
V_{CCP} — Programming supply voltage: Address	8.75	9.0	9.25	V
Protect		0		V
Maximum time V_{CCP} >5.25V			1.0	Sec
Programming voltage: Address	8.75	9.0	9.25	V
Protect	8.75		9.25	V
Programming current: Address			5	mA
Protect			50	mA
t_r — Programming pulse rise time: Address	10		100	μS
Protect	10		100	μS
t_ω — Programming pulse width	0.5		1.0	mS

Step 4: Return V_{CC} to 0-volts. (Note. *If the programming of all address bits is completed in less than 1-second, V_{CC} can remain at 9.0-volts for the required interval of time.*)

Step 5: Steps 1 through 3 are applicable to the programming of each address bit that requires a high-level $\overline{IV}$ match.

Step 6: To verify that the address is properly programmed, return V_{CC} to +5V, set $\overline{IV0}$-$\overline{IV7}$ to the desired (inverted) binary address pattern, set $\overline{ME}$ = WC = Low and SC = MCLK = High. If there are no programming errors, subsequent data written from $\overline{IV4}$-$\overline{IV7}$ (WC = High) will appear inverted on UD4-UD7.

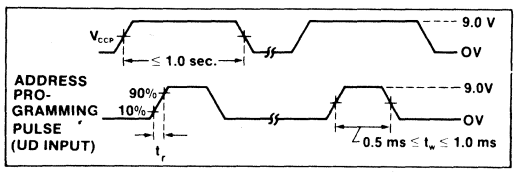


Figure 2. Address Programming Pulse

Address Protect

After programming the I/O Port, steps should be taken to isolate the address circuits and make these circuits permanently immune to further change.

Step 1: Set V_{CC} and all control inputs to 0-volts (V_{CC} = $\overline{UIC}$ = $\overline{UOC}$ = $\overline{ME}$ = SC = WC = MCLK = GND = 0.0V); $\overline{IV0}$-$\overline{IV7}$ = open circuit.

Step 2: Taking one pin at a time, apply a protect programming pulse (Figure 3) to each user-bus bit (UD0-UD7)—refer to Table 3 for min/max specifications pertaining to voltage and current.

Step 3: Verify that the address circuits for each bit is isolated by applying 9-volts, in turn, to each user-bus pin (UD0-UD7) and measuring less than 200 microamperes of input current. (Note. *Setup conditions are the same as those in Step 1.*)

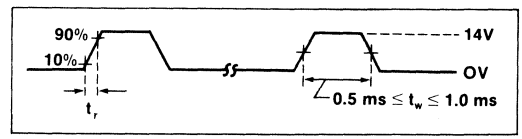


Figure 3. Protect Programming Pulse

DC ELECTRICAL CHARACTERISTICS

COMMERCIAL: 4.75V ≤ V_{CC} ≤ 5.25V, 0°C ≤ T_A ≤ 70°C
MILITARY: 4.5V ≤ V_{CC} ≤ 5.5V, -55°C ≤ T_C ≤ 125°C

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Power supply voltage[3]	+7	Vdc
V_{IN}	Input voltage[3]	+5.5	Vdc
T_{STG}	Storage temperature range	-65 to +150	°C

PARAMETER		TEST CONDITIONS	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{CC}	Supply Voltage		4.75	5	5.25	4.5	5	5.5	V
V_{IH}	High Level Input Voltage		2.0			2.0			V
V_{IL}	Low Level Input Voltage				0.8			0.8	V
V_{CL}	Input Clamp Voltage	V_{CC} = Min; I_I = -10mA			-1.5			-1.5	V
I_{IH}	High Level Input Current[1]	V_{CC} = Max; V_{IH} = 2.7V		5.0	100		5.0	100	μA
I_{IL}	Low Level Input Current[1]	V_{CC} = Max; V_{IL} = 0.5V		-350	-550		-350	-550	μA
I_{OZH}	High-Z State Output Current—High Level[4]	V_{CC} = Max; V_{OH} = 2.5V			100			100	μA
I_{OZL}	High-Z State Output Current—Low Level[4]	V_{CC} = Max; V_{OL} = 0.5V			-100			-100	μA
V_{OL}	Low Level Output Voltage $\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC} = Min; I_{OL} = 16mA			0.55			0.55	V
	User Bus (UD4-UD7)	V_{CC} = Min; I_{OL} = 24mA			0.55			0.55	V
V_{OH}	High Level Output Voltage	V_{CC} = Min; I_{OH} = -3.2mA	2.4			2.4			V
I_{OS}	Short Circuit Output Current[2] $\overline{IV}$ Bus ($\overline{IV0}$-$\overline{IV7}$)	V_{CC} = Max	-20			-20			mA
	UD Bus (UD4-UD7)	V_{CC} = Max	-10			-10			mA
I_{CC}	Supply Current	V_{CC} = Max; $\overline{ME}$ = $\overline{UOC}$ = V_{CC}		90	150		90	150	mA

Notes:
1. The input current includes the Three-state leakage current of the output driver on the data lines.
2. Only one output may be shorted at a time.
3. These limits do not apply during address programming.
4. Applies only to pins UD4-UD7.

AC ELECTRICAL CHARACTERISTICS

COMMERCIAL: 4.75V ≤ V_{CC} ≤ 5.25V, 0° C ≤ T_A ≤ 70° C

MILITARY: 4.5V ≤ V_{CC} ≤ 5.5V, -55° C ≤ T_C ≤ 125° C

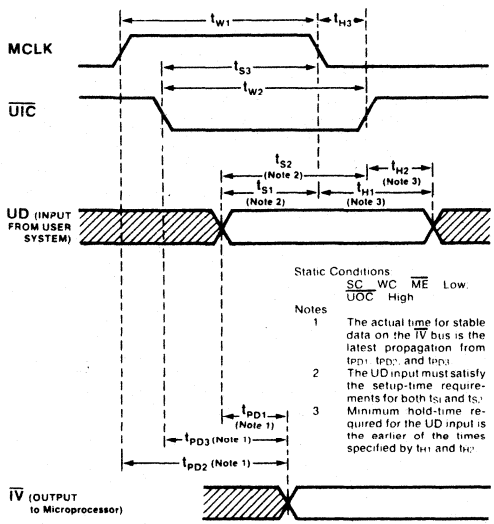
LOADING: See TEST LOADING CIRCUITS

PARAMETER	REFERENCES[1] FROM	REFERENCES[1] TO	TEST CONDITIONS	LIMITS (COMMERCIAL) Min	Typ	Max	LIMITS (MILITARY) Min	Typ	Max	UNIT
Pulse Widths:										
t_{W1} Clock High	↑MCLK	↓MCLK		35			35			ns
t_{W2} User Input Control	↓$\overline{UIC}$	↑$\overline{UIC}$	MCLK = High	35			35			ns
Propagation Delays:										
t_{PD1} UD Propagation Delay	UD_{0-3}	$\overline{IV}_{0-3}$	MCLK = High SC = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			30			30	ns
t_{PD2} UD Clock Delay	↑MCLK	$\overline{IV}_{0-3}$	UD_{0-3} = Stable; MCLK = High; SC = WC = $\overline{ME}$ = $\overline{UIC}$ = Low			50			50	ns
t_{PD3} UD Input Delay	↓$\overline{UIC}$	$\overline{IV}_{0-3}$	UD_{0-3} = Stable; MCLK = High; SC = WC = $\overline{ME}$ = Low			50			50	ns
t_{PD4} $\overline{IV}$ Data Propagation Delay	$\overline{IV}_{4-7}$	UD_{4-7}	MCLK = WC = High; $\overline{ME}$ = $\overline{UOC}$ = SC = Low			45			45	ns
t_{PD5} $\overline{IV}$ Data Clock Delay	↑MCLK	UD_{4-7}	$\overline{IV}_{4-7}$ = Stable; WC = High; $\overline{ME}$ = $\overline{UOC}$ = SC = Low			55			55	ns
Output Enable Timing:										
t_{OE1} UD Output Enable	↓$\overline{UOC}$	UD_{4-7}				30			30	ns
t_{OE3} $\overline{IV}$ Data Master Enable	↓$\overline{ME}$	$\overline{IV}$	WC = SC = Low			22			25	ns
t_{OE5} $\overline{IV}$ Data Write Recovery	↓WC	$\overline{IV}$	SC = $\overline{ME}$ = Low			25			25	ns
t_{OE6} $\overline{IV}$ Data Select Recovery	↓SC	$\overline{IV}$	WC = $\overline{ME}$ = Low			25			25	ns
Output Disable Timing:										
t_{OD1} UD Output Disable	↑$\overline{UOC}$	UD_{4-7}				25			25	ns
t_{OD3}[2] $\overline{IV}$ Data Master Disable	↑$\overline{ME}$	$\overline{IV}$	WC = SC = Low			25			25	ns
t_{OD5}[2] $\overline{IV}$ Data Write Override	↑WC	$\overline{IV}$	SC = $\overline{ME}$ = Low			20			20	ns
t_{OD6}[2] $\overline{IV}$ Data Select Override	↑SC	$\overline{IV}$	WC = $\overline{ME}$ = Low			20			20	ns
Setup Times:										
t_{S1} UD Clock Setup Time	UD_{0-3}	↓MCLK	$\overline{UIC}$ = Low	15			15			ns
t_{S2} UD Control Setup Time	UD_{0-3}	↑$\overline{UIC}$	MCLK = High	15			15			ns
t_{S3} User Input Control Setup Time	↓$\overline{UIC}$	↓MCLK		25			25			ns
t_{S4} $\overline{IV}$ Data Setup Time	$\overline{IV}$	↓MCLK	WC = High or SC = High; $\overline{ME}$ = Low;	35			35			ns
t_{S5}[3] $\overline{IV}$ Master Enable Setup Time	↓$\overline{ME}$	↓MCLK	WC = High or SC = High	30			30			ns
t_{S6} $\overline{IV}$ Write Control Setup Time	↑WC	↓MCLK	SC = $\overline{ME}$ = Low;	30			30			ns
t_{S7} $\overline{IV}$ Select Control Setup Time	↑SC	↓MCLK	WC = $\overline{ME}$ = Low	30			30			ns

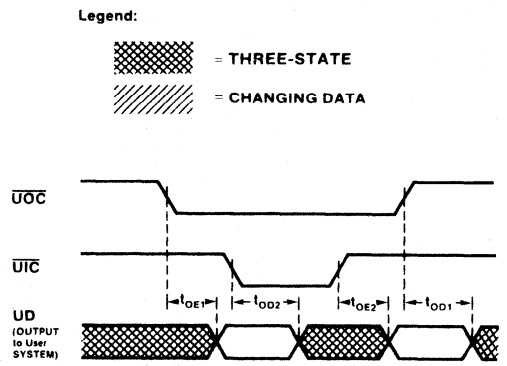
AC ELECTRICAL CHARACTERISTICS (Cont'd)

PARAMETER		REFERENCES FROM	REFERENCES TO	TEST CONDITIONS	LIMITS (COMMERCIAL) Min	Typ	Max	LIMITS (MILITARY) Min	Typ	Max	UNIT
Hold Times:											
t_{H1}	UD Clock Hold Time	↓MCLK	UD	$\overline{UIC}$ = Low	15			15			ns
t_{H2}	UD Control Hold Time	↑$\overline{UIC}$	UD	MCLK = High	15			15			ns
t_{H3}	User Input Control Hold Time	↓MCLK	↑$\overline{UIC}$		0			0			ns
t_{H4}	Data Hold Time	↓MCLK	$\overline{IV}$	$\overline{WC}$ = High or SC = High; $\overline{ME}$ = Low	5			5			ns
t_{H5}[3]	Master Enable Hold Time	↓MCLK	↑$\overline{ME}$	WC = High or SC = High;	0			0			ns
t_{H6}	$\overline{IV}$ Write Control Hold Time	↑MCLK	↓WC	SC = $\overline{ME}$ = Low	0			0			ns
t_{H7}	$\overline{IV}$ Select Control Hold Time	↑MCLK	↓SC	WC = $\overline{ME}$ = Low	0			0			ns

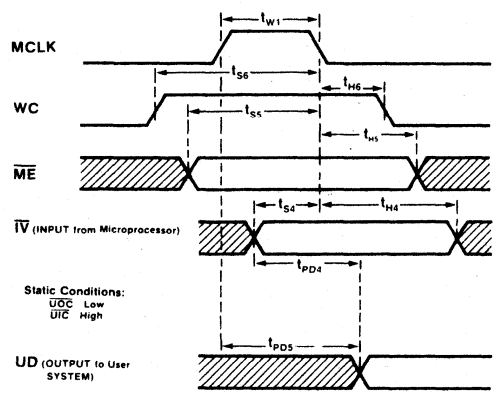
Notes:
1. All measurements to the $\overline{IV}$ bus assumes the address selection latch is set.
2. These parameters are measured with a capacitive loading of 50 *pf* and represent the output driver turn-off time.
3. If $\overline{ME}$ is to be high (inactive), it must be setup before the rising edge and held after the falling edge of MCLK to avoid unintended writing into or selection of the I/O port.



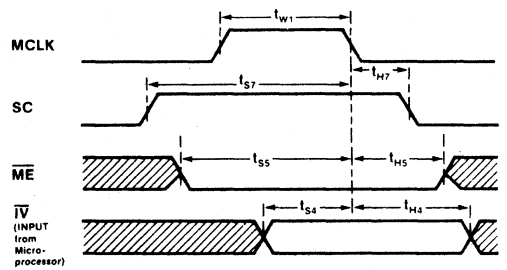
a. User Data Input Timing



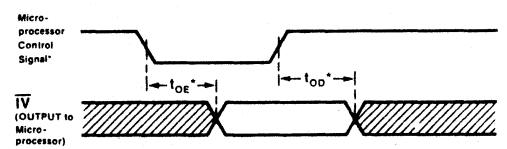
b. User Data Output Timing



c. MicroController Write Cycle Timing



d. MicroController Select Cycle Timing



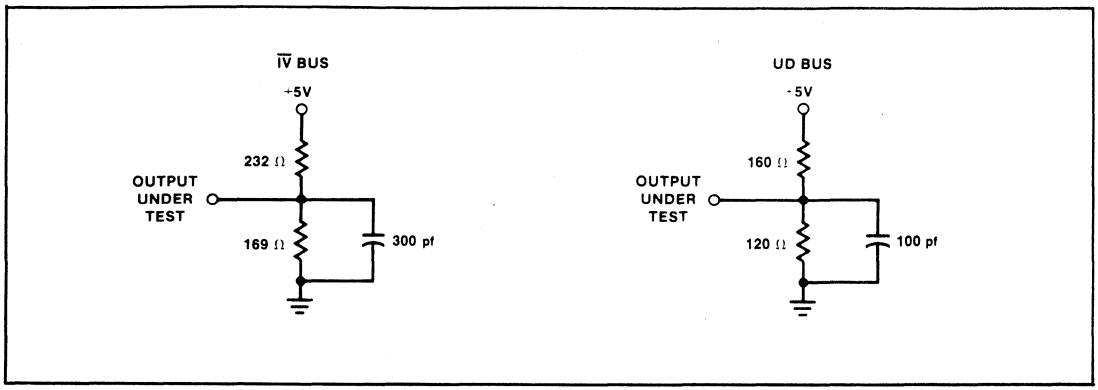
*PARAMETER KEY

MICROPROCESSOR CONTROL SIGNAL	AC TIMING PARAMETERS	STATIC CONDITIONS
$\overline{ME}$	t_{OE3} t_{OD3}	SC = WC = LOW
WC	t_{OE5} t_{OD5}	SC = $\overline{ME}$ = LOW
SC	t_{OE6} t_{OD6}	WC = $\overline{ME}$ LOW

e. MicroController Output Enable Timing

Figure 4. Timing Diagrams

TEST LOADING CIRCUITS



APPLICATIONS

When compared to other MicroController ports in the 8X370 series, the 8X382 has some unique features that provide real design advantages in certain applications. Connection of the I/O port to the MicroController is simple and straightforward in that like pin names are tied together. The system designer must also decide on which bank of the MicroController to place the 8X382 and then connect the $\overline{ME}$ pin of the port to either the $\overline{LB}$ (Left Bank) or $\overline{RB}$ (Right Bank) of the MicroController.

The 8X382 is unique because it can be used for both dedicated input and output operations. In the system shown below, the user interface requires nine (9) dedicated inputs and eleven (11) dedicated outputs. Observe that by using an 8X382, the problem is solved by three devices, whereas, four 8X372 ports are required for the same solution.

Another important use of the 8X382 is in implementing a handshake interface. Since both input and output bits reside in the same port, I/O operations can be performed without port re-addressing. Users may also find the 8X382 an advantage in the layout of Printed Circuit boards, since random control/status signals can be grouped within the same device position.

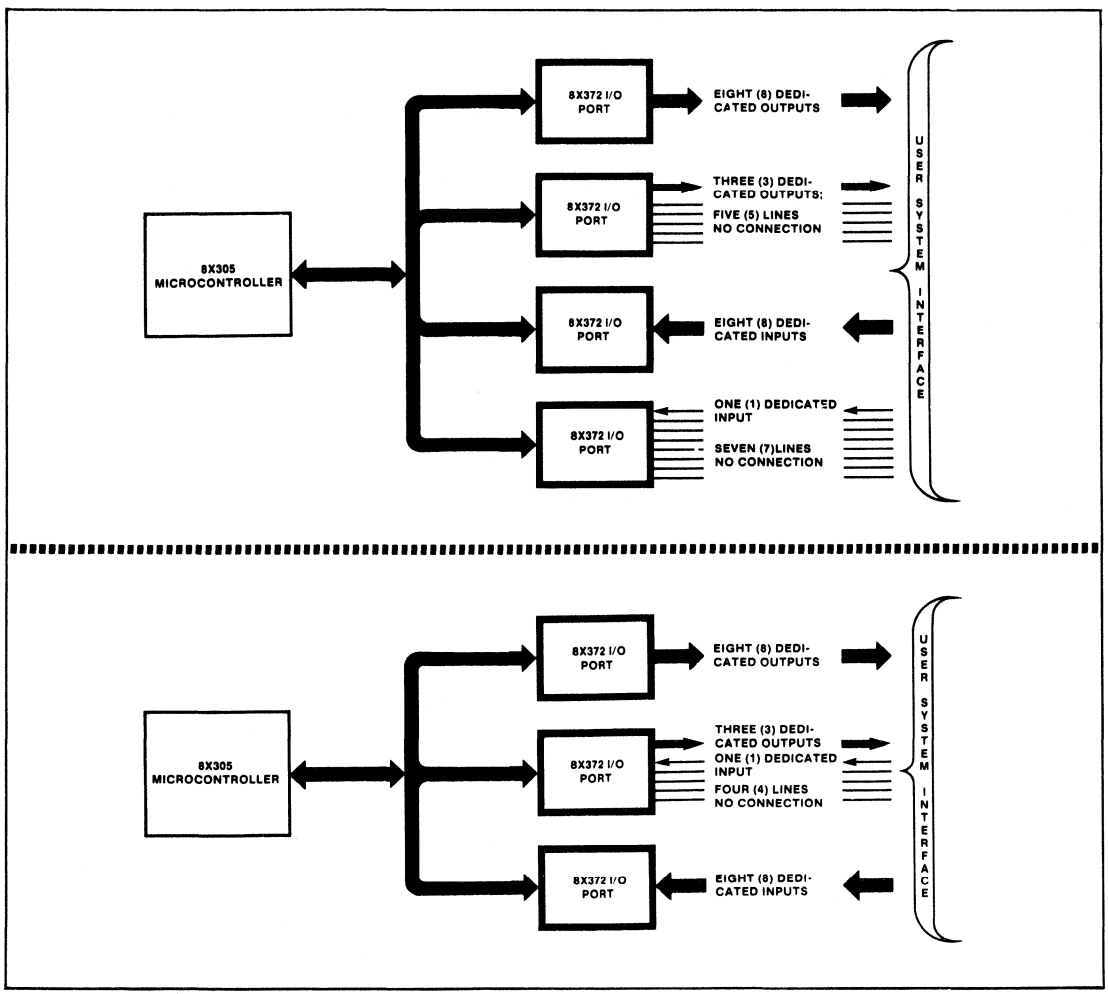


Figure 5. Logic Diagram for 8X382 I/O Port

8-BIT LATCHED BIDIRECTIONAL I/O PORT

Originally published by Signetics January 1984

FEATURES

- **Dual bidirectional ports**
- **Independent port operation (User-port priority for data entry)**
- **User data input synchronous**
- **At power-up, User-port outputs are high and Microprocessor-port outputs are high-Z**
- **Three-state TTL outputs for high-drive capabilities**
- **Directly compatible with 8X300 Microcontroller**
- **Single +5V supply**

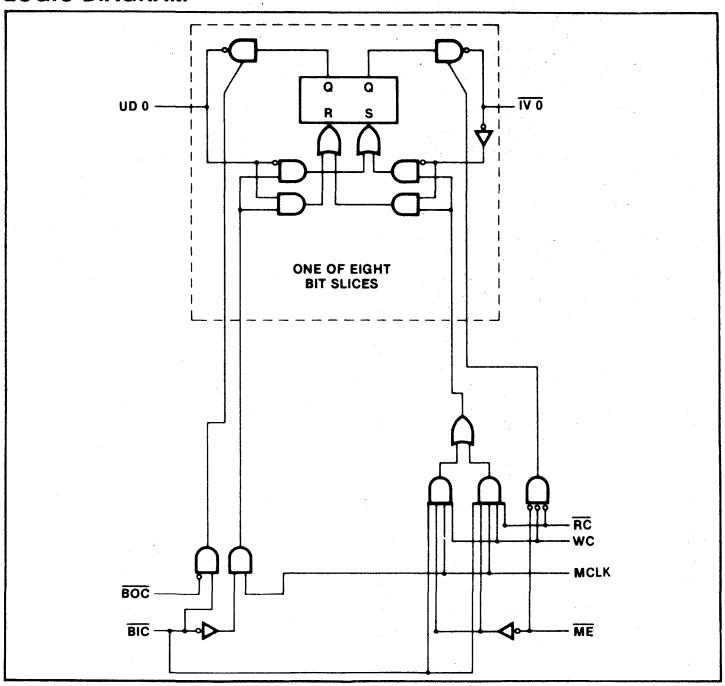
PRODUCT DESCRIPTION

The 8T31 is an 8-bit bidirectional data register designed to function as Input/Output interface elements in microprocessor systems.

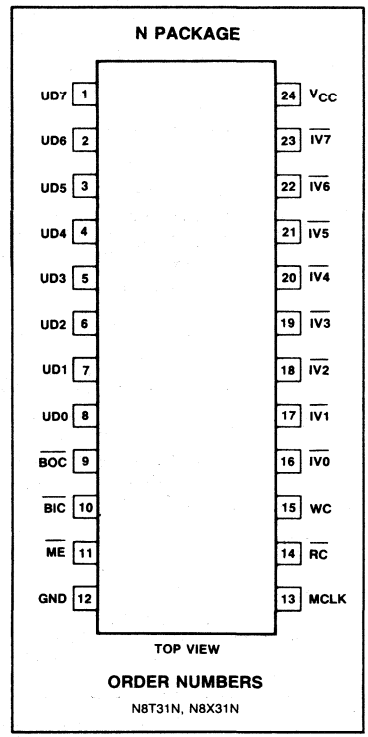
Each part contains eight clocked data latches that are accessible from either a *microprocessor* port or a *user* port. Separate I/O control is provided for each port. The two ports operate independently, except that when both are attempting to input data into the data latches, the User port (UD0-UD7) has priority. The master enable (ME) signal enables or disables the microprocessor bus regardless of the state of the other inputs but has no effect on the user bus.

A unique feature of these parts is their ability to start up in a predetermined state. If the clock is maintained at a level of less than 0.8 volts until the power supply reaches 3.5 volts, all bits of the user port will wakeup at a "logic 1" level and those of the microprocessor port will wakeup in the high-impedance state.

LOGIC DIAGRAM



PIN CONFIGURATION



ORDER NUMBERS

N8T31N, N8X31N

PIN DESIGNATION

PIN	SYMBOL	NAME AND FUNCTION	TYPE
1-8	UD0-UD7:	User Data I/O Lines. Bidirectional data lines to communicate with user's equipment.	Acitve high three-state
16-23	$\overline{IV0}$-$\overline{IV7}$:	Microprocessor Bus. Bidirectional data lines to communicate with controlling digital system.	Active low three-state
10	$\overline{BIC}$:	Input Control. User input to control writing into the I/O Port from the user data lines.	Active low
9	$\overline{BOC}$:	Output Control. User input to control reading from the I/O Port onto the user data lines.	Active low
11	$\overline{ME}$:	Master Enable. System input to enable or disable all other system inputs and outputs. It has no effect on user inputs and outputs.	Active low
15	WC:	Write Command. When WC is high, stores contents of $\overline{IV0}$-$\overline{IV7}$ as data.	Active high
14	$\overline{RC}$	Read Command. When RC is low, data is presented on $\overline{IV0}$-$\overline{IV7}$.	Active low
13	MCLK:	Master Clock. Input to strobe data into the latches. See function tables for details.	Active high
24	V_{CC}:	5V power connection.	
12	GND:	Ground.	

Table 1. USER PORT CONTROL FUNCTION

$\overline{BIC}$	$\overline{BOC}$	MCLK	USER DATA BUS FUNCTION
H	L	X	Output Data
L	X	H	Input Data
H	H	X	Inactive

H = High Level L = Low Level X = Don't care

Table 2. MICROPROCESSOR PORT CONTROL FUNCTION

$\overline{ME}$	$\overline{RC}$	WC	MCLK	$\overline{BIC}$	MICROPROCESSOR BUS FUNCTION
L	L	L	X	X	Output Data
L	X	H	H	H	Input Data
X	H	L	X	X	Inactive
X	X	H	X	L	Inactive
H	X	X	X	X	Inactive

USER DATA BUS CONTROL

The activity of the user data bus is controlled by the $\overline{BIC}$ and $\overline{BOC}$ inputs as shown in Table 1.

The user data input is a synchronous function with MCLK. A low level on the $\overline{BIC}$ input allows data on the user data bus to be written into the data latches only if MCLK is at a high level. A low level on the BIC input allows data on the user data bus to be latched regardless of the level of the MCLK input.

To avoid conflicts at the data latches, input from the microprocessor port is inhibited when BIC is at a low level. Under all other conditions the 2 ports operate independently.

MICROPROCESSOR BUS CONTROL

As is shown in Table 2, the activity of the microprocessor port is controlled by the $\overline{ME}$, RC, WC and $\overline{BIC}$ inputs, as well as the state of an internal status latch. $\overline{BIC}$ is included to show user port priority over the microprocessor port for data input.

BUS OPERATION

Data written into the 8T31 from one port will appear inverted when read from the other port. Data written into the 8T31 from one port will not be inverted when read from the same port.

DC ELECTRICAL CHARACTERISTICS

V_{CC} = 5V ± 5%, 0°C ≤ T_A ≤ 70°C unless otherwise specified.

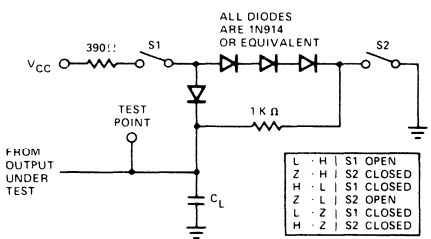
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage:						V
V_{IH}	High		2.0			
V_{IL}	Low				.8	
V_{IC}	Clamp	I_I = 5mA			-1	
Output voltage:		V_{CC} = 4.75V				V
V_{OH}	High		2.4			
V_{OL}	Low				.55	
Input current[1]:		V_{CC} = 5.25V				μA
I_{IH}	High	V_{IH} = 5.25V		<10	100	
I_{IL}	Low	V_{IL} = .5V		-350	-550	
Output current[2]:						mA
I_{OS}	Short circuit	V_{CC} = 4.75V				
	UD bus		10			
	IV bus		20			
I_{CC}	VCC supply current	V_{CC} = 5.25V		100	150	mA

NOTES

1. The input current includes the three-state/open collector leakage current of the output driver on the data lines.
2. Only one output may be shorted at a time.

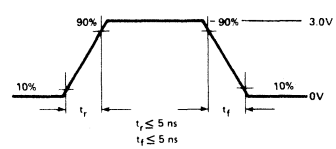
PARAMETER MEASUREMENT INFORMATION

LOAD CIRCUIT FOR THREE-STATE OUTPUTS

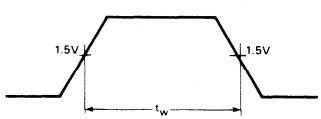


NOTE: C_L includes fixture capacitance

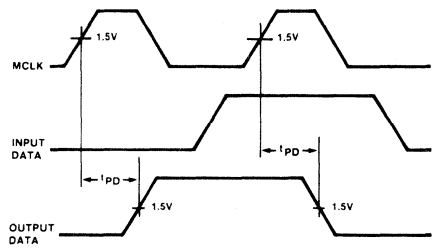
INPUT WAVEFORM



CLOCK PULSE WIDTH



DATA DELAY TIMES
Clock Referenced



AC ELECTRICAL CHARACTERISTICS $0°C \leq T_A \leq 70°C$, $V_{CC} = 5V \pm 5\%$

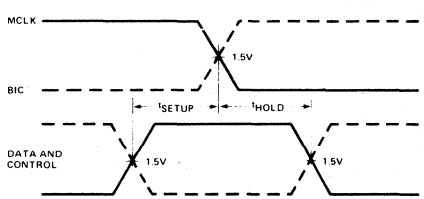
PARAMETERS		INPUT	TEST CONDITIONS	LIMITS			UNIT
				Min	Typ	Max	
		UD X			25	38	ns
t_{PD}	User data relay[1]	MCLK	C_L = 50pF		45	61	ns
t_{OE}	User output enable	$\overline{BOC}$	C_L = 50pF	18	26	47	ns
t_{OD}	User output disable	$\overline{BIC}$	C_L = 50pF	18	28	35	ns
		BOC		16	23	33	ns
t_{PD}	µP data delay[1]	$\overline{IV}$ X	C_L = 50pF		38	53	ns
		MCLK			48	61	ns
t_{OE}	µP output enable	$\overline{ME}$ $\overline{RC}$ WC	C_L = 50pF	14	19	25	ns
t_{OD}	µP output disable	$\overline{ME}$ $\overline{RC}$ WC	C_L = 50pF	13	17	32	ns
t_W	Minimum pulse width	MCLK		40			ns
t_{SETUP}	Minimum setup time[2]	$\overline{UD X}$[3]		15			ns
		$\overline{BIC}$		25			
		$\overline{IV}$ X		55			
		$\overline{ME}$		30			
		$\overline{RC}$		30			
		WC		30			
t_{HOLD}	Minimum hold time[2]	$\overline{UD X}$[3]		25			ns
		$\overline{BIC}$		10			
		$\overline{IV}$ X		10			
		$\overline{ME}$		5			
		$\overline{RC}$		5			
		WC		5			

NOTES

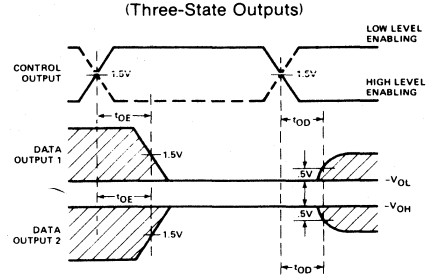
1. Data delays referenced to the clock are valid only if the input data is stable at the arrival of the clock and the hold time requirement is met.
2. Set up and hold times given are for "normal" operation. $\overline{BIC}$ setup and hold times are for a user write operation. $\overline{RC}$ setup and hold times are for an I/O Port select operation. $\overline{ME}$ and WC setup and hold times are for a microprocessor bus write operation.
3. Times are referenced to MCLK.

VOLTAGE WAVEFORMS

SETUP AND HOLD TIMES



OUTPUT ENABLE AND DISABLE TIMES
(Three-State Outputs)



WAVEFORM #1 IS FOR AN OUTPUT WITH INTERNAL CONDITIONS SUCH THAT THE OUTPUT IS LOW WHEN THE THREE-STATE DRIVER IS ENABLED. WAVEFORM #2 IS FOR THE OPPOSITE CONDITION.

8-BIT LATCHED ADDRESSABLE BIDIRECTIONAL I/O PORT

Originally published by Signetics January 1984

FEATURES

- **Independent port operation (user-port priority for data entry)**
- **User data input available as synchronous (8T32) or as asynchronous (8T36)**
- **User data bus available with three-state (8T32, 8T36)**
- **At power-up, user-port outputs are high and microprocessor-port outputs are high-z; status latch (from address compare) is also cleared at power-up**
- **Three-state TTL outputs for high-drive capabilities**
- **Directly compatible with 8X300 microcontroller**
- **Single +5V supply**

PRODUCT IDENTITY

8T32— Three-state, field-programmable (addresses 0-255), synchronous user port.

8T36—Three-state, field-programmable (addresses 0-255), asynchronous user port

PRODUCT DESCRIPTION

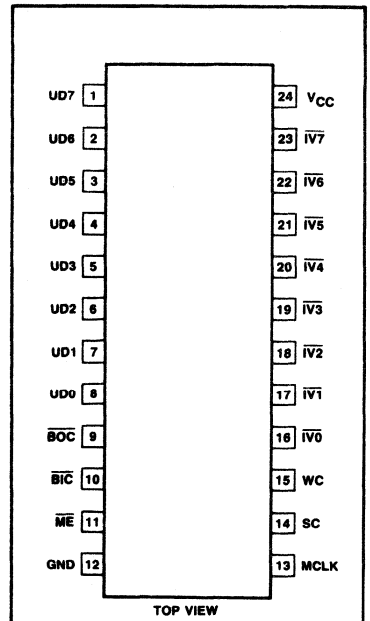
8T32/8T36. Each of these I/O Bytes is an addressable and bi-directional register designed for use as an interface element in any system with TTL-compatible buses. (*Note.* Since these I/O Bytes are frequently used with the 8X300 Microcontroller and its associated Interface Vector bus, the 8T32-8T36 family of parts are commonly called IV Bytes.) Each I/O Byte contains eight identical data latches (Bits 0 through 7); the latches are accessed from either of two 8-bit ports—one port connecting to the microprocessor (8X300) and the other port connecting to the user device.

Separate controls are provided for each port and the two ports operate independently, except when both attempt to input data at the same time; in this case, the user port bus has priority.

The address of each I/O Byte is field-programmable and the microprocessor port is accessed when a valid address is received; the user port is accessible at all times. A selected Byte is automatically deselected when the address of another I/O Byte is sensed on the address/data bus. A Master Enable (ME) input is available for use as a ninth address bit, allowing direct access to 512 I/O Bytes without address decoding.

A unique feature of these parts is their ability to start up in a predetermined state. If the clock is maintained at a level of less than 0.8 volts until the power supply reaches 3.5 volts, all bits of the user port will wakeup at a "logic 1" level and those of the microprocessor port will wakeup in the high-impedance state.

PIN CONFIGURATION



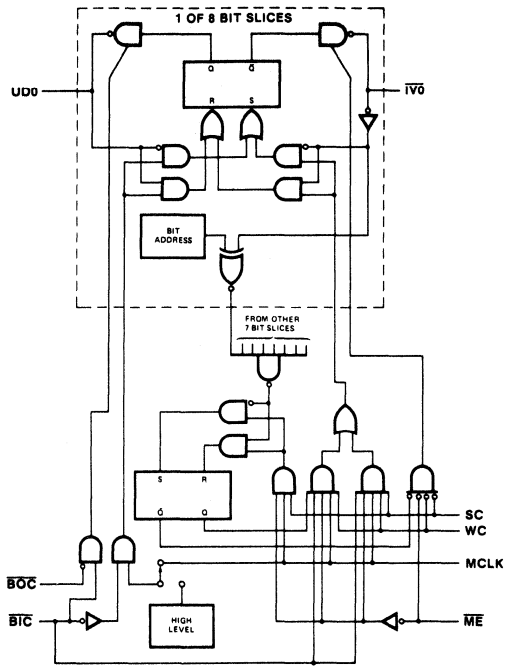
TOP VIEW

ORDER NUMBERS

CONTACT LOCAL SALES REPRESENATIVE

A stock of 8T32s and 8T36s with addresses "1" through "10" are maintained in inventory; with a longer lead time, a small quantity of address "11" through "50" are also available.

TYPICAL BLOCK DIAGRAM



*Switch indicates synchronous/asynchronous user write option. Switch shown for synchronous version.

PIN DESCRIPTION

PIN	SYMBOL	NAME AND FUNCTION	TYPE
1-8	UD0-UD7:	User Data I/O Lines. Bidirectional data lines to communicate with user's equipment. Either tri-state or open collector outputs are available.	Active high
16-23	$\overline{IV0}$-$\overline{IV7}$:	Microprocessor Bus. Bidirectional data lines to communicate with controlling digital system (microprocessor).	Active low three-state
10	$\overline{BIC}$:	Input Control. User input to control writing into the I/O Port from the user data lines.	Active low
9	$\overline{BOC}$:	Output Control. User input to control reading from the I/O Port onto the user data lines.	Active low
11	$\overline{ME}$:	Master Enable. System input to enable or disable all other system inputs and outputs. It has no effect on user inputs and outputs.	Active low
15	WC:	Write Command. When WC is high and SC is low, I/O Port, if selected, stores contents of $\overline{IV0}$-$\overline{IV7}$ as data.	Active high
14	SC:	Select Command. When SC is high and WC is low, data on $\overline{IV0}$-$\overline{IV7}$ is interpreted as an address. I/O Port selects itself if its address is identical to μP bus data; it de-selects itself otherwise.	Active high
13	MCLK:	Master Clock. Input to strobe data into the latches. See function tables for details.	Active high
24	VCC:	5V power connection.	
12	GND:	Ground.	

Table 1. USER PORT CONTROL FUNCTION

$\overline{BIC}$	$\overline{BOC}$	MCLK	USER DATA BUS FUNCTION	
			8T32	8T36
H	L	X	Output Data	Output Data
L	X	H	Input Data	Input Data
L	X	L	Inactive	Input Data
H	H	X	Inactive	Inactive

H = High Level L = Low Level X = Don't care

Table 2. MICROPROCESSOR PORT CONTROL FUNCTION

$\overline{ME}$	SC	WC	MCLK	$\overline{BIC}$	STATUS LATCH	I/O PORT FUNCTION
L	L	L	X	X	SET	Output Data
L	L	H	H	H	SET	Input Data
L	H	L	H	X	X	Input Address
L	H	H	H	L	X	Input Address
L	H	H	H	H	X	Input Data and Address
L	X	H	L	X	X	Inactive
L	H	X	L	X	X	Inactive
L	L	H	H	L	X	Inactive
L	L	X	X	X	Not Set	Inactive
H	X	X	X	X	X	Inactive

USER DATA BUS

The activity of the user data bus is controlled by the $\overline{BIC}$ and $\overline{BOC}$ inputs as shown in Table 1.

For the 8T32, user data input is a synchronous function with MCLK. A low level on the $\overline{BIC}$ input allows data on the user data bus to be written into the data latches only if MCLK is at a high level. For the 8T36, user data input is an asynchronous function. A low level on the $\overline{BIC}$ input allows data on the user data bus to be latched regardless of the level of the MCLK input. Note that when the 8T36, is used with the 8X300 Microcontroller, care must be taken to insure that the Microprocessor bus is stable when it is being read by the 8X300 Microcontroller.

To avoid conflicts at the Data Latches, input from the Microprocessor Port is inhibited when $\overline{BIC}$ is at a low level. Under all other conditions the two ports operate independently

MICROPROCESSOR BUS CONTROL

As is shown in Table 2, the activity of the microprocessor port is controlled by the $\overline{ME}$, SC, WC, and $\overline{BIC}$ inputs, as well as the state of an internal status latch. $\overline{BIC}$ is included to show user port priority over the microprocessor port for data input.

Each I/O Port's status latch stores the result of the most recent I/O Port select; it is set when the I/O Port's internal address matches the Microprocessor Bus. It is cleared when an address that differs from the internal address is presented on the Microprocessor Bus.

In normal operation, the state of the status latch acts like a master enable; the microprocessor port can transfer data only when the status latch is set.

When SC and WC are both high, data on the Microprocessor Bus is accepted as data, whether or not the I/O Port was selected. The data is also interpreted as an address. The I/O Port sets its select status if its address matches the data read when SC and WC were both high; it resets its select status otherwise.

BUS OPERATION

Data written into the I/O Port from one port will appear inverted when read from the other port. Data written into the I/O Port from one port will not be inverted when read from the same port.

AC ELECTRICAL CHARACTERISTICS $0°C \le T_A \le 70°C$, $V_{CC} = 5V \pm 5\%$

PARAMETER	INPUT	TEST CONDITION	LIMITS			UNIT
			Min	Typ	Max	
t_{PD} User data delay (Note 1)	UD X	C_L = 50pF		25	38	ns
	MCLK*			45	61	
	$\overline{BIC}$†			40	55	
t_{OE} User output enable	$\overline{BOC}$	C_L = 50pF	18	26	47	ns
t_{OD} User output disable	$\overline{BIC}$	C_L = 50pF	18	28	35	ns
	$\overline{BOC}$		16	23	33	
t_{PD} μP data delay (Note 1)	$\overline{IV\ X}$	C_L = 50pF		38	53	ns
	MCLK			48	61	
t_{OE} μP output enable	$\overline{ME}$	C_L = 50pF	14	19	25	ns
	SC					
	WC					
t_{OD} μP output disable	$\overline{ME}$	C_L = 50pF	13	17	32	ns
	SC					
	WC					
t_W Minimum pulse width	MCLK		40			ns
	$\overline{BIC}$†		35			
t_{SETUP} Minimum setup time	UD X□	(Note 2)	15			ns
	$\overline{BIC}$*		25			
	$\overline{IV\ X}$		55			
	$\overline{ME}$		30			
	SC		30			
	WC		30			
t_{HOLD} Minimum hold time	UD X□	(Note 2)	25			ns
	$\overline{BIC}$*		10			
	$\overline{IV\ X}$		10			
	$\overline{ME}$		5			
	SC		5			
	WC		5			

• Applies for 8T32.
† Applies for 8T36.
□ Times are referenced to MCLK for 8T32, and are referenced to BIC for 8T36.

NOTES:

1. Data delays referenced to the clock are valid only if the input data is stable at the arrival of the clock and the hold time requirement is met.
2. Set up and hold times given are for "normal" operation. $\overline{BIC}$ setup and hold times are for a user write operation. SC setup and hold times are for I/O Port select operation. $\overline{ME}$ setup and hold times are for both IV write and select operations.

DC ELECTRICAL CHARACTERISTICS $0°C \leq T_A \leq 70°C$, $V_{CC} = 5V \pm 5\%$

PARAMETER		TEST CONDITIONS	LIMITS			UNITS
			Min	Typ	Max	
V_{IH}	High-level input voltage		2.0		5.5	V
V_{IL}	Low-level input voltage		-1.0		.8	V
V_{CL}	Input clamp voltage	$I_I = -5mA$			-1.0	V
I_{IH}	High-level input current[1]	$V_{CC} = 5.25V$ $V_{IH} = 5.25V$		<10	100	μA
I_{IL}	Low level input current[1]	$V_{CC} = 5.25V$ $V_{IL} = .5V$		-350	-550	μA
V_{OL}	Low-level output voltage	$V_{CC} = 4.75V$ $I_{OL} = 16mA$			.55	V
V_{OH}	High-level output voltage	$V_{CC} = 4.75V$ $I_{OH} = -3.2mA$	2.4			V
I_{OS}	Short-circuit output current[2] UD bus IV bus	 $V_{CC} = 4.75V$ $V_{CC} = 4.75V$	 10 20			 mA mA
I_{CC}	Supply current	$V_{CC} = 5.25V$		100	150	mA

NOTES

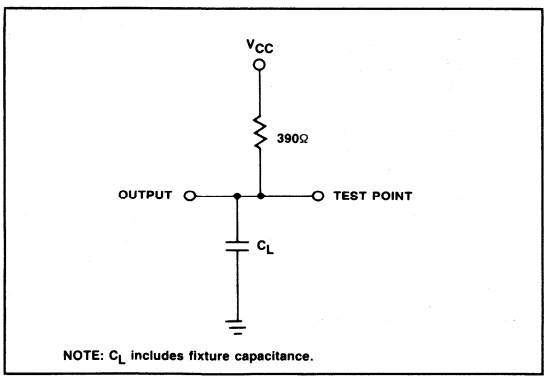
1. The input current includes the Three-state/Open Collector leakage current of the output driver on the data lines.
2. Only one output may be shorted at a time.
3. These limits do not apply during address programming.

Absolute Maximum Ratings:

Supply voltage[3] 7V

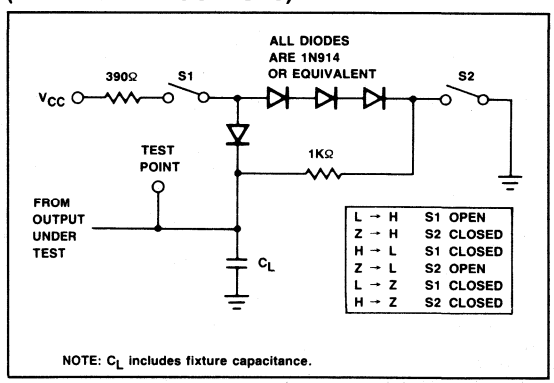
Input voltage[3] 5.5V

TEST LOAD CIRCUIT (OPEN COLLECTOR OUTPUTS)



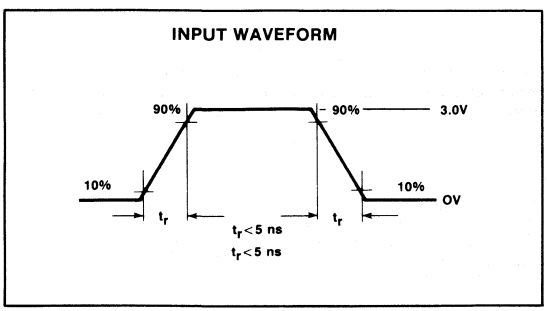
NOTE: C_L includes fixture capacitance.

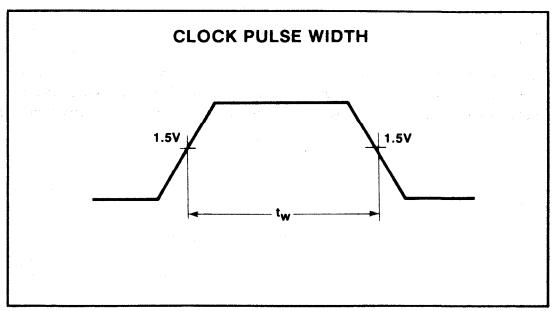
TEST LOAD CIRCUIT (THREE-STATE OUTPUTS)



NOTE: C_L includes fixture capacitance.

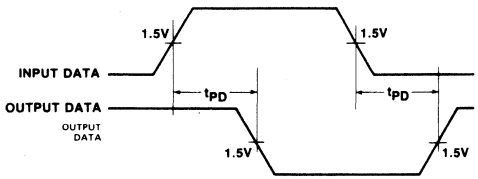
VOLTAGE WAVEFORMS



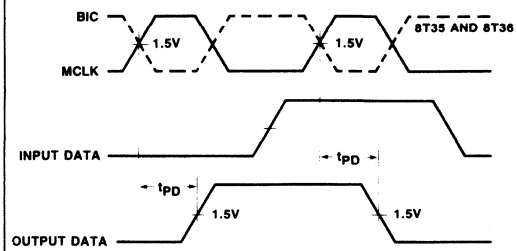


VOLTAGE WAVEFORMS (Cont'd)

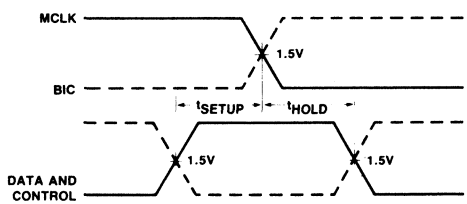
DATA DELAY TIMES
Input Data Reference



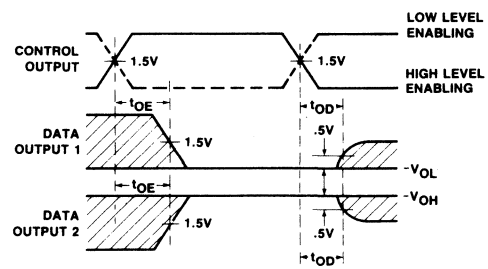
DATA DELAY TIMES
Clock Referenced



SETUP AND HOLD TIMES



OUTPUT ENABLE AND DISABLE TIMES
(Three-State Outputs)



Waveform #1 is for an output with internal conditions such that the output is Low when the three-state driver is enabled. Waveform #2 is for the opposite condition.

ADDRESS PROGRAMMING

The I/O Port is manufactured such that an address of all high levels (>2V) on the Microprocessor Bus inputs matches the Port's internal address. To program a bit so a low-level input (<0.8V) matches, the following procedure should be used:

1. Set all control inputs to their inactive state ($\overline{BIC}$ = $\overline{BOC}$ = $\overline{ME}$ = V_{CC}, SC = WC = MCLK = GND). Leave all Microprocessor Bus I/O pins open.
2. Raise V_{CC} to 7.75V ± .25V.
3. After V_{CC} has stabilized, apply a single programming pulse to the user data bus bit where a low-level match is desired. The voltage should be limited to 18V; the current should be limited 75mA. Apply the pulse as shown in Figure 1.
4. Return V_{CC} to 0V. (Note 1).
5. Repeat this procedure for each bit where a low-level match is desired.
6. Verify that the proper address is programmed by setting the Port's status latch ($\overline{IV0}$-$\overline{IV7}$ = desired address, $\overline{ME}$ = WC = L, SC = MCLK = H). If the proper address has been programmed, data presented at the μP bus will appear inverted on the user bus outputs. (Use normal V_{CC} and input voltage for verification.)

After the desired address has been programmed, a second procedure must be followed to isolate the address circuitry. The procedure is:

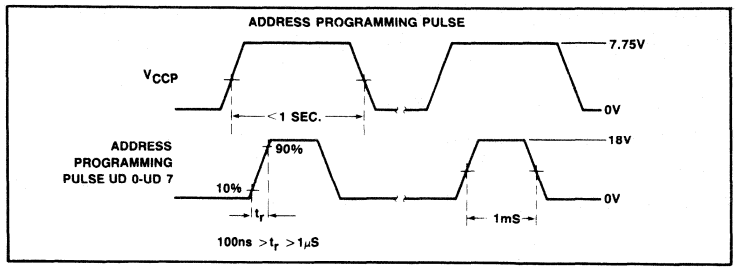


Figure 1

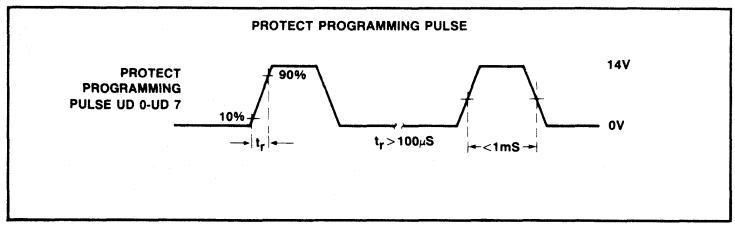


Figure 2

1. Set V_{CC} and all control inputs to 0V. (V_{CC} = $\overline{BIC}$ = $\overline{BOC}$ = $\overline{ME}$ = SC = WC = MCLK = 0V). Leave all Microprocessor Bus I/O pins open.
2. Apply a protect programming pulse to every user data bus pin, one at a time. The voltage should be limited to 14V; the current should be limited to 150mA. Apply the pulse as shown in Figure 2.
3. Verify that the address circuitry is isolated by applying 7V to each user data bus pin and measuring less than 1mA of input current. The conditions should be the same as in step 1 above. The rise time on the verification voltage must be slower than 100μs.

PROGRAMMING SPECIFICATIONS[1]

PARAMETER		TEST CONDITIONS	LIMITS			UNITS
			Min	Typ	Max	
V_{CCP}	Programming supply voltage					
	Address		7.5		8.0	V
	Protect			0		V
I_{CCP}	Programming supply current	V_{CCP} = 8.0V			250	mA
	Max time V_{CCP}>5.25V				1.0	s
	Programming voltage					
	Address		17.5		18.5	V
	Protect		13.5		14.0	V
	Programming current					
	Address				75	mA
	Protect				150	mA
	Programming pulse rise time					
	Address		.1		1	μs
	Protect		100			μs
	Programming pulse width		.5		1	ms

NOTE

1. If all programming can be done in less than 1 second, V_{CC} may remain at 7.75V for the entire programming cycle.

APPLICATIONS

Figure 3 shows some of the various ways to use the I/O Port in a system. By controlling the $\overline{BIC}$ and $\overline{BOC}$ lines, the device may be used for the input and output of data, control, and status signals. I/O Port 1 functions bidirectionally for data transfer and I/O Port 2 provides a similar function for discrete status and control lines. I/O Ports 3 and 4 serve as dedicated output and input ports, respectively.

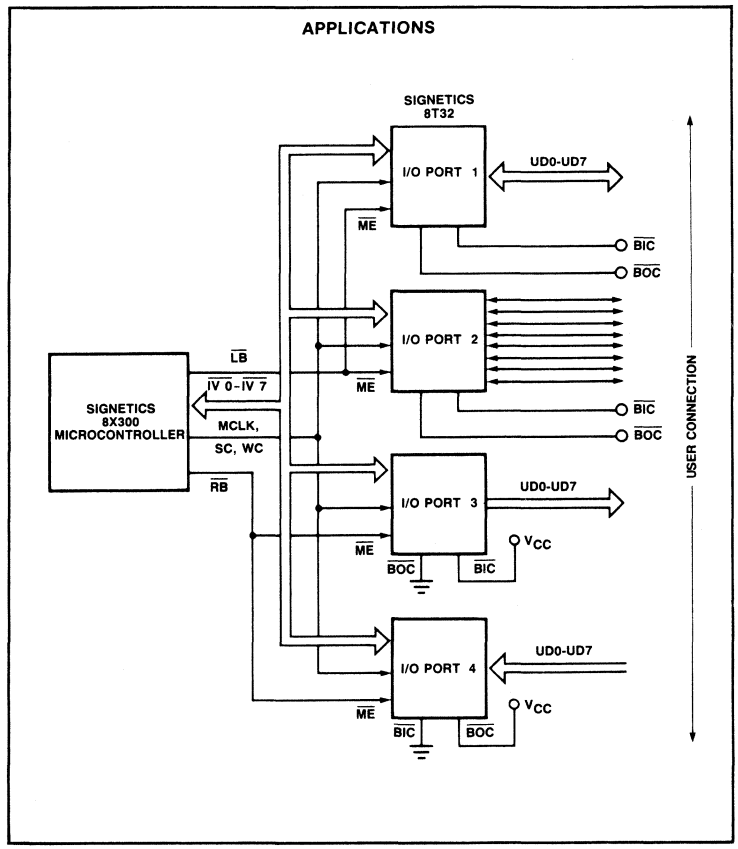


Figure 3

Product support

PRODUCT SUPPORT

Originally published by Signetics January 1984

SUPPORT FACILITIES

The 8X300 Family is strongly supported with Development Systems, Support Software, Applications, Training and Documentation. Together, this support provides the user with a powerful set of tools to evaluate, design, debug, and implement a simple or complex system.

DEVELOPMENT SYSTEMS

EZ-PRO (Manufactured by American Automation)
- Universal Development System
- Relocating macroassembler
- Full speed in-circuit emulator/debugger
- Maximum memory support

8X305 ICEPACK (Manufactured by Sigen Corp.)
- Full speed in-circuit emulator/debugger
- RS-232 interface to CP/M or Intellec
- 4K word memory/8-bit extended microcode
- Low cost

Signetics 8X305 Prototyping System
- Single board module
- RS-232 interface
- Resident monitor
- 256 to 4096 words of Writable Control Storage
- Minimum cost

SOFTWARE

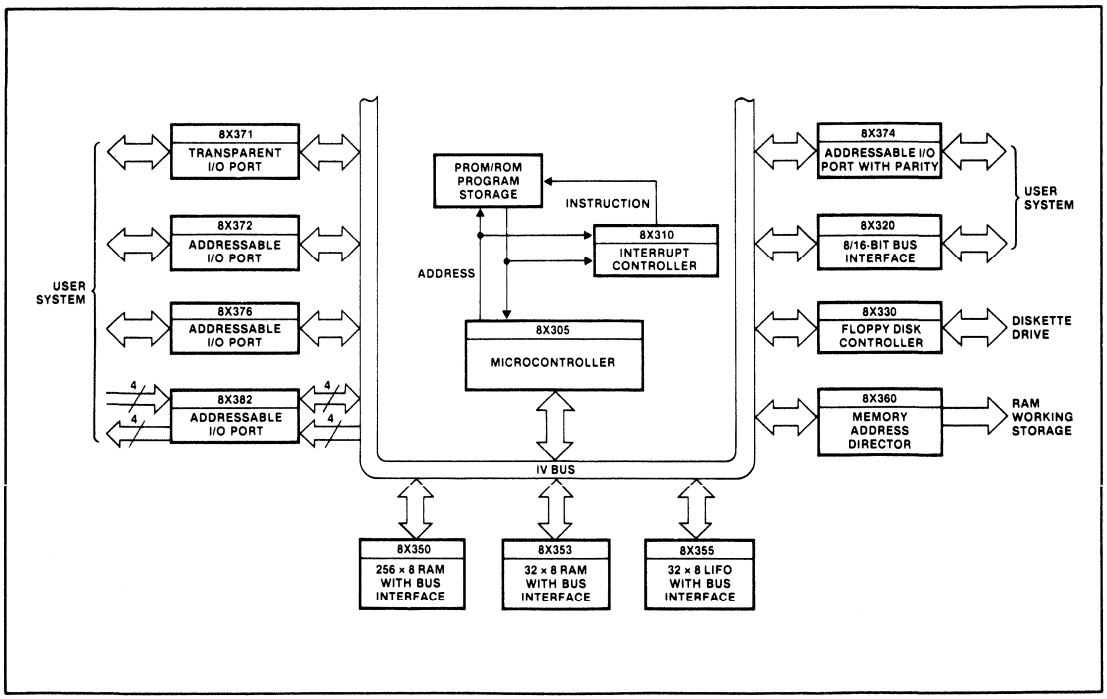
MCCAP CrossAssembler
- Full function macroassembler
  - — Free format source code
  - — Symbolic address assignment
  - — Nested macro support
  - — Cross-reference table
  - — Supports extended microcode
- Multiple output formats
- Available in two versions
  - — FORTRAN source code
  - — Intellec/ISIS object code

APPLICATIONS SUPPORT

Field applications engineers
Product applications engineers
Application notes
- Floppy disk controller
- ECC
- Hard disk controller
- Local network interfacing

TRAINING AND DOCUMENTATION

- Videocassette training course
- Designer's seminar
- 8X305 User's Manual
- 8X300 Family Product Capabilities Manual
- MCCAP Programming Manual
- Full complement of Data Sheets



PREFACE

Originally published by Signetics January 1984

8X305 Prototyping System

This document provides the information required to understand, set up and operate the 8X305 Prototyping System. It is recommended that the user become thoroughly familiar with the 8X305 MicroController and the high speed peripherals that support the device. For this purpose, the following documents are recommended:

- 8X305 Users Manual — comprehensive functional detail, interface characteristics, hardware and software design data, and systems information for the 8X305 MicroController.
- 8X300 Family Product Capabilities Manual — overview of the 8X300 Family of parts, including application information on the 8X305 MicroController and its support devices.
- MCCAP Manual — complete description of the powerful MicroController Cross-Assembler Program for the 8X300 and 8X305.
- Data Sheets — electrical and functional characteristics for each member of the 8X300 Family and related parts.
  - 8X305 MicroController
  - 8X310 Interrupt Control Coprocessor
  - 8X320 Bus Interface Register Array
  - 8X330 Floppy Disk Formatter/Controller
  - 8X338 Local Area Network Controller
  - 8X350 256 Byte Bipolar RAM
  - 8X360 Memory Address Director
  - 8X371 Latched 8-bit Bidirectional I/O Port
  - 8X372/8X376 Addressable 8-bit Bidirectional I/O Port
  - 8X374 Addressable 8-bit Bidirectional I/O Port with Parity
  - 8X382 Addressable 4-IN/4-OUT I/O Port
  - 8X60 FIFO RAM Controller

INTRODUCTION

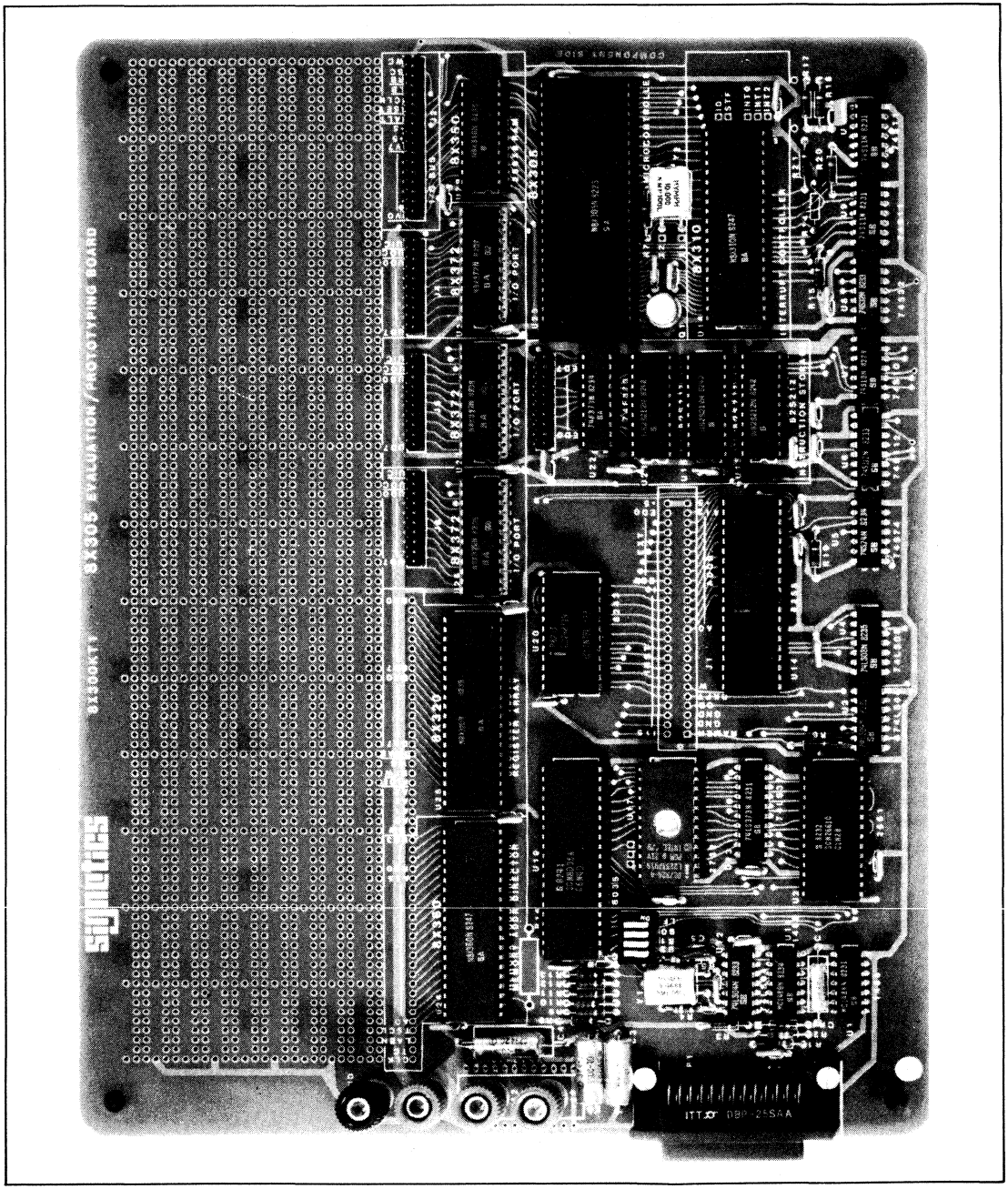


Figure 1-1. 8X305 Prototyping System

1.1 SYSTEM DESCRIPTION

The 8X305 Prototyping System is a powerful design support tool that aids the engineer in the evaluation, design, and prototyping of systems based on the Signetics 8X305 MicroController and its family of support devices. Its advanced features permit the development of both 8X305 firmware and application circuitry. The prototyping systems capabilities are adequate to serve as a complete development system for simple systems and provide a low-cost tool for evaluating portions of more complex designs.

1.2 ARCHITECTURAL OVERVIEW

As shown in Figure 1-1, the 8X305 Prototyping System consists of a single printed circuit board that includes an 8X305 MicroController and various 8X305 Family peripheral devices. The 8X305's microprogram resides in Writeable Control Storage (WCS). A Monitor Processor controls operation of the 8X305 by loading the WCS, activating the Run/ Step logic, and directly placing instructions onto the 8X305's Instruction bus. The Monitor Processor also controls the User Interface, which is through a standard RS-232 connector. The remainder of the board is occupied by power connections and a large wire-wrap area for prototyping of user-developed circuits. A complete discussion of the operation and interrelationship of these functions is contained in later chapters.

1.2.1 USER INTERFACE

The User Interface of the 8X305 Prototyping System is accomplished through a standard RS-232 connector. Data rates from 110 to 19,200 baud are switch selected by the user. The monitor program contained in the system controls all user communication, which is accomplished interactively through a straightforward and user-friendly syntax. While the operation of the system requires only a low-cost "dumb" terminal, it can be connected to a host computer to support more advanced developments. Commands are included to support up and down loading of programs in such applications.

1.2.2 MONITOR PROCESSOR AND RUN/STEP LOGIC

Operation of the system is controlled by the Monitor Processor, which is implemented using an 8035 Microprocessor. The Monitor Processor is responsible for the following functions:

- User interaction
- Loading Writeable Control Storage
- Loading and reading 8X305 registers and I/O devices
- Control of Run/Step logic

Programming for the Monitor Processor is supplied by Signetics and is contained in a PROM.

1.2.3 WRITEABLE CONTROL STORAGE

8X305 MicroController programs are executed from a Writeable Control Storage (WCS) that is contained on the board. The prototyping system is supplied with 256 words of instruction memory. An expansion module is available to support address space requirements of up to 4096 words. The WCS is sufficiently fast to permit full speed operation of the 8X305.

Writeable Control Storage words are 25 bits wide to support advanced microprogramming requirements. Sixteen of these bits contain actual 8X305 instructions. Eight of the remaining bits are used to support "Extended Microcode" designs as described in the 8X305 Users Manual. The 25th bit, transparent to the user, is set by the Monitor Processor to control breakpoints.

Since the three-bus architecture of the 8X305 does not permit the MicroController to modify its own program memory, the WCS is loaded by the Monitor Processor.

1.2.4 8X305 MICROCONTROLLER AND PERIPHERALS

An 8X305 MicroController and various 8X300 Family peripheral devices are included in the prototyping system.

The Instruction and Program Address busses of the 8X305 are connected to Writeable Control Storage (WCS) as well as an 8X310 Interrupt Control Coprocessor (ICC). The interrupt and status pins of the 8X310 are available to the user for use in prototyping real-time or other interrupt driven systems.

The 8X305's IV bus is connected to the following 8X300 Family peripheral devices:

(1) 8X320 Bus Interface Register Array
(1) 8X350 256 Byte Bipolar RAM
(1) 8X360 Memory Address Director
(3) 8X372 Addressable 8-bit Bidirectional I/O Ports

IV bus data and control signal connections are also available to the user to permit attachment of other devices or user developed logic. User interface connections to the 8X320, 8X360, and 8X372's are available adjacent to the wire-wrap area to permit prototyping of various 8X305 based designs.

SYSTEM SETUP

2.1 POWER CONNECTIONS

> *CAUTION*
>
> *Power connections must be properly made; otherwise, component damage will result.*

Power connections to the Prototyping System are made through four binding posts located on the left hand edge of the board. These are labeled GROUND, +5, —12, and +12. Without options or user circuitry in the wire-wrap area, the current drawn from each power source is as follows:

+5 VDC — less than 2.5 amperes
+12 VDC — less than 20 milliamperes
—12 VDC — less than 20 milliamperes

Additional current must be supplied for any options or user circuitry added to the board. At the user's option, a DC-to-DC converter (converts +5 VDC to ±12 VDC) can be installed in the space allotted to the —12 VDC and +12 VDC binding posts. Refer to the parts list in Appendix B for the manufacturer and part number of the recommended device.

2.2 RS-232 INTERFACE

An RS-232 connector is provided in the lower left-hand corner of the system for interconnection to the user's CRT terminal, and is connected as shown in Table 2-1.

Table 2-1 RS-232 CONNECTOR

Pin No.	Signal	Description
1	GND	Ground
2	TXD	Transmit Data(in)
3	RXD	Receive Data(out)
4	RTS	Request to Send(in)
5	CTS	Clear to Send(out)
6	DSR	Data Set Ready(out)
7	GND	Ground
8	CAD	Carrier Detect(out)

RS-232 specifications call for data communications equipment (DCE), such as this system, to be connected to data terminal equipment (DTE), such as the user's CRT terminal. When connecting this system to another DCE, such as a host computer, be sure to interchange TXD (pin 2) with RXD (pin 3), and RTS (pin 4) with CTS (pin 5). This can be done on the cable or it can be accomplished logically by using a Null Modem. Due to the variety of interpretations of RS-232, some terminals may not immediately work with the Prototyping System. If difficulties are encountered, first reduce the connections to three signals: TXD, RXD, and GND. Then if necessary, interchange TXD and RXD signals.

2.3 BAUD RATE SELECTION

Any of the eight baud rates listed in Table 2-2 may be selected by proper setting of switches B2, B1 and B0 located near the RS-232 connector.

Table 2-2 BAUD RATE

B2	B1	B0	Baud Rate
0	0	0	19200
0	0	1	9600
0	1	0	4800
0	1	1	2400
1	0	0	1200
1	0	1	600
1	1	0	300
1	1	1	110

(0 = Switch off, 1 = Switch on)

For hard-copy printing terminals, an optional line-feed feature may be selected to avoid over-strike of characters after a backspace on error. This feature is selected by setting the LF switch located next to switch B0 to the ON position.

2.4 EXTERNAL OSCILLATOR CONNECTIONS

> *CAUTION*
>
> *To prevent possible damage to the crystal, never apply an external oscillator signal to X1 or X2 input with crystal Y2 connected to the circuit.*

The Prototyping System is supplied with a 10 MHz crystal; therefore it operates the 8X305 MicroController at its full rated speed of 200 nanoseconds per instruction. The crystal may be changed by the user to any frequency from 4 MHz to 10 MHz. Alternatively, an external oscillator may be connected to the X1 and X2 inputs of the 8X305, as described in the 8X305 Users Manual. The external oscillator may operate at any frequency between 0.2 MHz and 10 MHz. Note that the 8X305 is capable of running at frequencies lower than 0.2 MHz, but the Prototyping System's Monitor Processor expects the 8X305 to be finished executing an instruction within 10 microseconds, thereby imposing a lower limit of 0.2 MHz. Tie points for X1 and X2 are located next to crystal Y2 and the 8X305. Be sure to disconnect crystal Y2 before connecting the external oscillator inputs to X1 and X2.

2.5 INHIBIT JUMPER FOR 8X310 ROM DISABLE

The 8X310 Interrupt Control Coprocessor connects to the Instruction and Address buses of the 8X305. It accomplishes interrupt and subroutine control by disabling the control storage that contains the 8X305's microprogram and placing specific JMP instructions onto the instruction bus. To permit the Prototyping System to operate either with or without an 8X310 in the circuit, a jumper is incorporated into the ROM Disable (RD) circuitry.

When the 8X310 Interrupt Control Coprocessor is physically present in location U16, the ROM Disable Inhibit Jumper located at R14 next to the 8X310 must not be present so that the 8X310 can disable WCS and avoid bus contention problems. When the 8X310 is not present, this jumper must be connected to the board at location R14 to permanently enable the Writeable Control Store RAMs.

SYSTEM OPERATION

3.1 POWER UP AND DIAGNOSTICS

When power is applied to the Prototyping System, resident diagnostic programs are executed to test the Micro-Controller and Writeable Control Store (WCS). The following message is then printed:

```
8X305 PROTOTYPING SYSTEM
REV n
SYSTEM CHECK ON
*
```

where "n" equals the current revision level.

If the Prototyping System is functioning improperly, either "MEMORY ERROR" or "8X305 ERROR" messages will be printed. Then the "prompt" character (*) will be printed and the user may examine WCS memory or 8X305 functions to diagnose the problem.

3.2 MONITOR PROGRAM

With the printing of the "prompt" character (*), system control passes to the monitor program. The user may then enter any of the ten monitor commands:

I **INPUT 8X305 instructions into memory:**
Accepts a starting WCS memory address and then permits the entering of an 8X305 instruction in mnemonic form and an extension instruction in octal notation.

n **Register examine/change, where "n" = register number:**
Displays the register number and its contents and allows a new value to be substituted.

(R0 may be accessed by its alternate name AUX by entering "A" or, R10 may be accessed by its alternate name OVF by entering "0".)

G **GO execute user's 8X305 program:**
Accepts a starting memory address and executes at full speed until a breakpoint or keyboard entry is reached. Then the contents of the registers and the next executable instruction are displayed, and single stepping can proceed.

S **STEP, single step user's program:**
Accepts a starting memory address and displays the contents of the registers and the instruction at that memory address. The instruction is executed by hitting the space bar; and successive instructions by successively hitting the space bar.

M **MEMORY and breakpoint examine/change:**
Accepts a starting WCS memory address and then displays the contents of that location in mnemonic form and indicates a possible breakpoint by an exclamation point. A breakpoint at this location may be set by typing an exclamation point or cleared by typing a backspace. A new 8X305 instruction and extension instruction is then entered by typing an I, as with the INPUT command.

L **LB, left-bank examine/change:**
Accepts a bank address (or the currently enabled address, if a blank is entered) and displays the contents of that left-bank address (0-377 octal). A new value may be entered to substitute for the original content.

R **RB, right-bank examine/change:**
Operates the same as the LB command except action is upon the right-bank.

D **DUMP memory contents to terminal:**
Accepts a starting WCS memory address and an ending address, and then dumps the memory contents from the start address to the end address onto the RS-232 port in ASCII HEX QUOTE format as described in Section 3.6.

F **FILL memory contents from terminal:**
Accepts a starting WCS memory address and fills memory in ASCII HEX QUOTE format as described in Section 3.6.

X **XCODE, temporarily set extended microcode latch:**
Accepts a new extension code value to be temporarily set in the extended microcode latch for control of user circuitry. The content of the extended microcode section of WCS is not changed by this command.

Although the user need only enter a single letter command, the monitor will respond by typing the whole command name as indicated in capitals above.

Any of the commands may be aborted before completion by typing an ASCII "control-C" character. While entering any number (sequence of octal digits), corrections may be made by entering a backspace character and then entering the correct number.

3.3 COMMAND SYNTAX DIAGRAMS

System commands and monitor responses are defined in the syntax diagrams in Figure 3-1a, b and c.

3.4 SAMPLE USAGE

The sample program in Figure 3-3 is shown to give the user an idea of a typical session and includes most commands used by the Prototyping System. A short program is input to WCS via the terminal Keyboard that continually increments R6 of the 8X305 and writes each new (incremented) value to location 133 of the 8X350 on the Right-Bank of the IV Bus and to I/O Port 001 on the Left-Bank. Since extended microcode is not needed in this example, none was entered as indicated by "/000".

3.5 BREAKPOINTS

Breakpoints are designed to halt the 8X305 just prior to the execution of an instruction on which a breakpoint has been specified. If the GO command is issued to start at an address that has a breakpoint set, the system will not stop on that address immediately. If the 8X305 should access that address again then a breakpoint stop will occur.

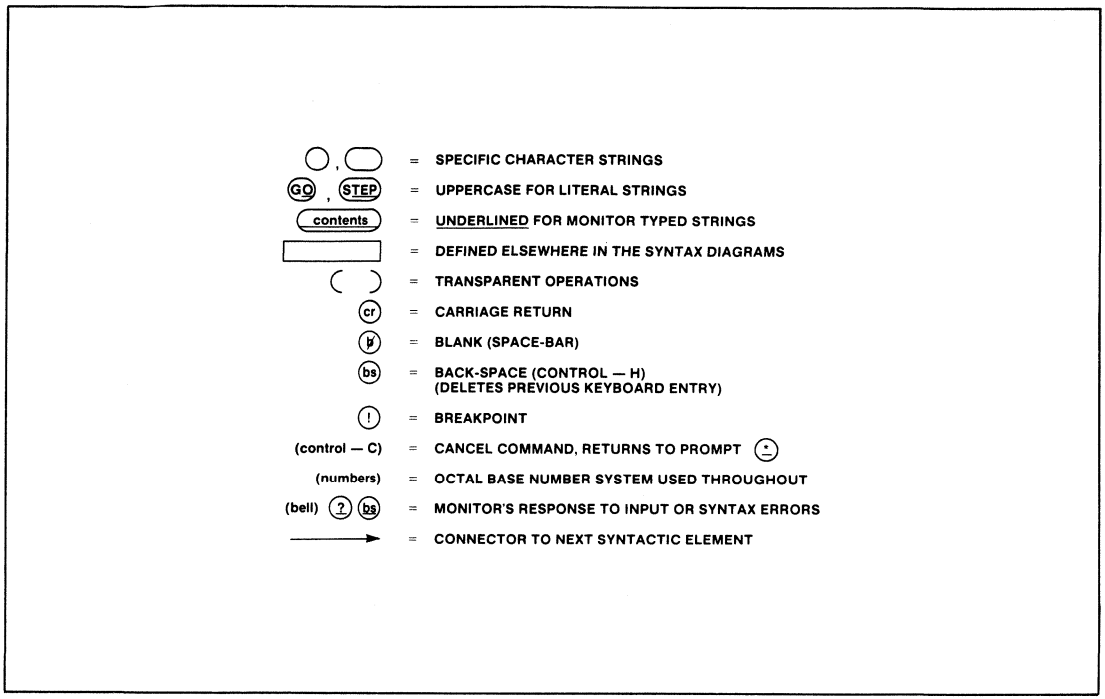


Figure 3-1a. Syntax Definitions

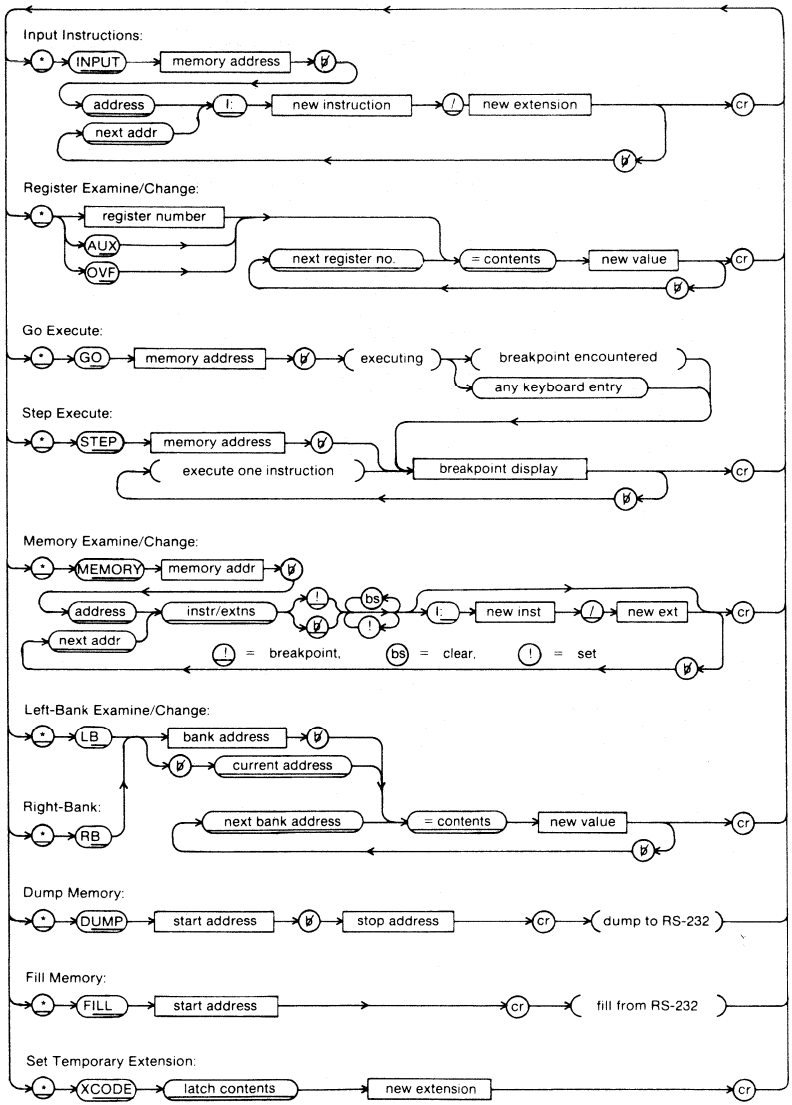
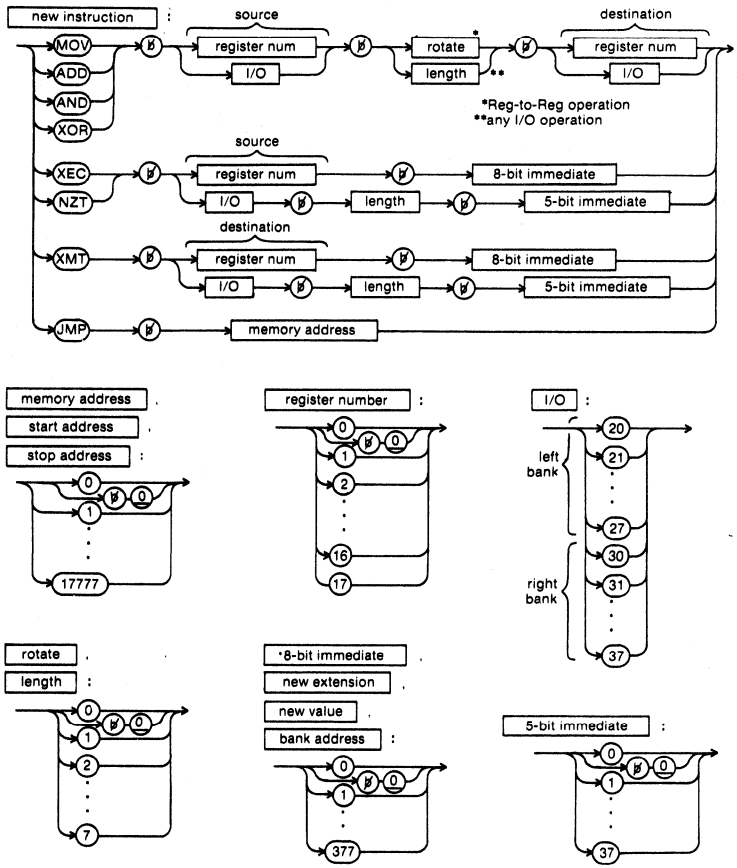


Figure 3-1b. Syntax Flow



note: — First two lines are column headers, printed each breakpoint or each 20 single steps.
— Third line is register contents.
— Fourth line is breakpoint; address — instruction/extension. This is the next instruction to be executed in single step.

Figure 3-1c. Syntax Flow

```
*
* DUMP 00000 00014
DUMPING HIGH BYTES ....
(STX)
00'C0'24'B9'C0'AC'68'74'01'10'70'11'F1'
(ETX)
DUMPING LOW BYTES ....
(STX)
00'00'08'BF'10'FF'FF'96'06'FF'7D'40'D7'
(ETX)
DUMPING EXT BYTES ....
(STX)
00'00'00'00'00'50'00'00'00'00'00'00'00'
(ETX)
*
*
```

Figure 3-2. Example of ASCII HEX QUOTE Format

3.6 ASCII HEX QUOTE FORMAT

Memory contents to and from the terminal during the DUMP and FILL commands are in an object code format commonly supported by PROM programming hardware known as "ASCII HEX QUOTE" format. Each block of eight-bit wide data is preceeded by an STX (ASCII Start of Text) character. Then each 8-bit byte is represented by 2 ASCII HEX characters (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) followed by a single quote ('). As many bytes as necessary may be transmitted until an ETX (ASCII End of Text) character terminates the block. Three such blocks are transmitted for any specified memory range: high byte of 8X305 instruction, low byte of 8X305 instruction, and extended microcode byte. Figure 3-2 shows an example of ASCII HEX QUOTE format.

3.7 XEC (EXECUTE) INSTRUCTIONS

When stepping through a user program and an 8X305 XEC instruction is encountered, the system will display an "XR" after the next instruction to indicate an XEC range of address.

Note that stepping must start on or before an XEC instruction for the program flow to be correct. If while running a program, a breakpoint or stop occurs on an instruction that is the target of an XEC instruction, an "XR" will not be displayed and program flow will not be correct if single stepping is then begun.

It is valid to nest any number of XEC instructions. Single stepping will work properly provided that the user start on or before the first XEC instruction.

3.8 8X310 INTERRUPT CONTROL COPROCESSOR CONSIDERATIONS

Certain 8X305 "NOP" instructions have specific meanings to the 8X310 as indicated in the data sheet. When using an 8X310 in the system, the user must start running or stepping from an instruction that is not an 8X310 instruction. It is valid, however, to start running and encounter a breakpoint or stop on an 8X310 instruction and then continue single stepping the program.

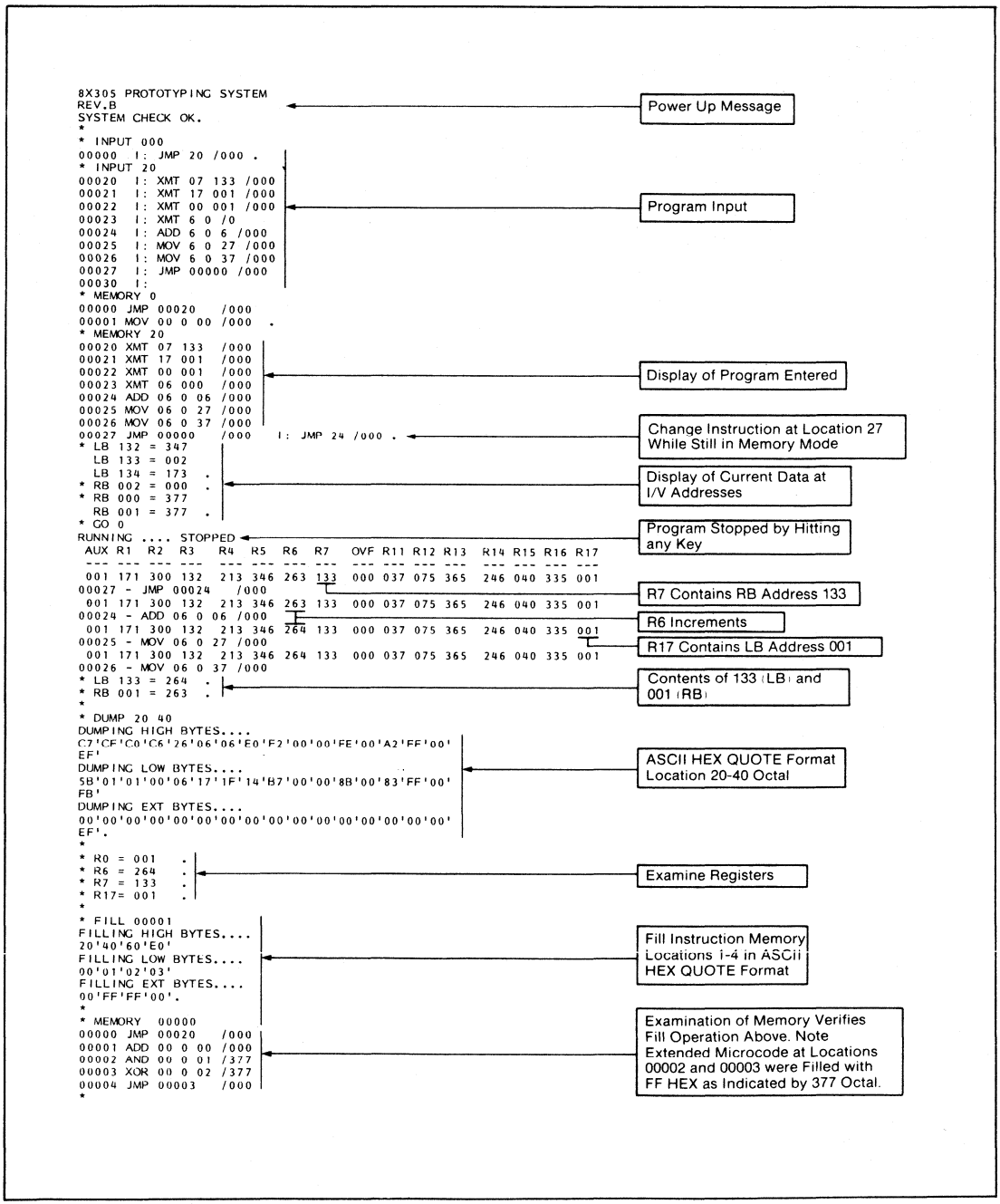


Figure 3-3. Sample Usage

USER CONNECTIONS

4.1 WIRE-WRAP AREA

The wire-wrap area located at the top of the board provides 26 square inches for user prototyping. This space accommodates standard IC widths from 0.3 to 0.9 inches and also provides power and ground connections.

4.2 IV BUS CONNECTIONS

The IV bus is the major communications link between the 8X305 MicroController and its family of peripherals. The bus is logically partitioned into two banks, referred to as the Left-Bank and Right-Bank. In the Prototyping System the 8X350 Working Storage RAM is connected to the Left-Bank, and the 8X320 Register Array, 8X360 Memory Address Director, and 8X372 I/O Ports are connected to the Right-Bank. All IV bus signals are present at connector J2 as shown in Table 4-1.

Table 4-1 IV BUS CONNECTOR J2

Pin No.	Signal	Description
1	$\overline{IV0}$	BUS (MSB)
3	$\overline{IV1}$	BUS
5	$\overline{IV2}$	BUS
7	$\overline{IV3}$	BUS
9	$\overline{IV4}$	BUS
11	$\overline{IV5}$	BUS
13	$\overline{IV6}$	BUS
15	$\overline{IV7}$	BUS(LSB)
17	V_{CC}	+5 V
19	V_{CC}	+5 V
21	$\overline{HALT}$	Halt
23	$\overline{RESET}$	Reset
25	MCLK	Clock
27	$\overline{LB}$	Left-Bank
29	$\overline{RB}$	Right-Bank
31	SC	Select Control
33	WC	Write Control
(All even pins)	GND	Ground

4.3 I/O PORTS

These 8X372 I/O ports have been provided on the Right-Bank of the IV bus for latching of input or output data. The ports are programmed for addresses 000, 001 and 002 and all signals are available at connectors J4, J5 and J6 respectively. Signals on these connectors are described in Table 4-2. Note that I/O port compatibility allows the user to substitute an 8X376 or 8X382 for any one of the 8X372 I/O ports. (One 8X371 non-addressable I/O port may be substituted, but ONLY if all other components on that bank are removed.)

Table 4-2 I/O PORT CONNECTORS J4, J5, AND J6

Pin No.	Signal	Description
2	UD7	User Data (LSB)
4	UD6	User Data
6	UD5	User Data
8	UD4	User Data
10	UD3	User Data
12	UD2	User Data
14	UD1	User Data
16	UD0	User Data (MSB)
18	$\overline{UOC}$	Output Control
20	UIC	Input Control
(All odd pins)	GND	Ground

4.4 EXTENDED MICROCODE CAPABILITY

"Extended Microcode" is a technique commonly used in 8X305 MicroController based designs to optimize performance. It is implemented by designing program memory to be wider than the 16-bit instruction word required for the 8X305. The additional bits are referred to as the extension and can be used for fast I/O selection or other system control and status monitoring purposes.

The Prototyping System uses a 24-bit instruction word, thus providing facilities for 8 bits of extended microcode. These bits are accessible to the user at connector J3 as shown in Table 4-3.

Table 4-3 EXTENDED MICROCODE BITS AT CONNECTOR J3

Pin No.	Signal	Description
1		No Connection
3	ED0	Extended Microcode (MSB)
5	ED1	Extended Microcode
7	ED2	Extended Microcode
9	ED3	Extended Microcode
11	ED4	Extended Microcode
13	ED5	Extended Microcode
15	ED6	Extended Microcode
17	ED7	Extended Microcode (LSB)
19		No Connection
(All even pins)	GND	Ground

Table 4-4 8X320 SIGNAL CONNECTIONS

Signal	Description
B/$\overline{W}$	Byte/Word Control
A0	Primary Port Address (LSB)
A1	Primary Port Address
A2	Primary Port Address
A3	Primary Port Address (MSB)
$\overline{PIOE}$	Programmed I/O Enable
R/$\overline{W}$	Read/Write Control
WS	Write Strobe
$\overline{DMAE}$	Direct Mem. Access Enable
D7A	Primary Data Port (LSB)
D6A	Primary Data Port
D5A	Primary Data Port
D4A	Primary Data Port
D3A	Primary Data Port
D2A	Primary Data Port
D1A	Primary Data Port
D0A	Primary Data Port
D7B	Primary Data Port
D6B	Primary Data Port
D5B	Primary Data Port
D4B	Primary Data Port
D3B	Primary Data Port
D2B	Primary Data Port
D1B	Primary Data Port
D0B	Primary Data Port (MSB)

4.5 8X310 CONNECTIONS

The 8X310 is connected to the 8X305 MicroController and the Writeable Control Store RAM to provide interrupt and subroutine capability. Five additional signals are provided for user interface as described in the 8X310 data sheet. These signals are accessible at tie points just to the right of the 8X310 chip:

STF	Stack Full Status
ID	Interrupt Disable Control
INT0	Interrupt 0 Input
INT1	Interrupt 1 Input
INT2	Interrupt 2 Input

4.6 8X320 CONNECTIONS

An 8X320 Bus Interface Register Array has ben provided on the IV Bus as a Right-Bank I/O device for interfacing to the user's system. The primary data, status and command signals are accessible at tie points located between the 8X320 and the wire-wrap area. A list of these signals is provided in Table 4-4.

4.7 8X360 CONNECTIONS

The 8X360 Memory Address Director has been incorporated into the Prototyping System design to facilitate implementation of a DMA channel. It is connected to the IV Bus as a Right-Bank I/O device. Interconnection to the signals listed in Table 4-5 can be made at the tie points located between the 8X360 and the wire-wrap area.

In applications using extended microcode to enable I/O devices care must be taken to avoid IV Bus contention with 8X300 Family peripheral devices enabled through the more commonly used address — select cycle. Refer to the 8X305 Users Manual for more information on extended microcode operations.

4.8 MEMORY EXPANSION

The Prototyping System is provided with 256 24-bit words of Writeable Control Storage; 16 bits for 8X305 instructions and 8 bits for extended microcode. The depth of Control Storage can be increased by the connection of an expansion module to connector J1, as shown in Table 4-6. A 4096 word Writeable Control Storage Expansion Module is available from Signetics (Part Number 8X300KT2SK). A schematic for this expansion module is provided in Appendix D.

Table 4-5 8X360 SIGNAL CONNECTIONS

Signal	Description
CLK	Clock Input
TC	Terminal Count Status
LABN	Loop Abort Control
TSCL	Tri-state Control
A0	Address Output (LSB)
A1	Address Output
A2	Address Output
A3	Address Output
A4	Address Output
A5	Address Output
A6	Address Output
A7	Address Output
A8	Address Output
A9	Address Output
A10	Address Output
A11	Address Output
A12	Address Output
A13	Address Output
A14	Address Output
A15	Address Output (MSB)
RS0	Register Select (LSB)
RS1	Register Select
RS2	Register Select
RS3	Register Select (MSB)

Table 4-6 MEMORY EXPANSION CONNECTOR J1

Pin No.	Signal	Description	Pin No.	Signal	Description
1	RAMEN	Disables on-board RAM	2	I15	8X305 Instruction (LSB)
3	GND	Ground	4	I14	8X305 Instruction
5	GND	Ground	6	I13	8X305 Instruction
7	OD	Output Disable	8	I12	8X305 Instruction
9	BKPT	Breakpoint	10	I11	8X305 Instruction
11	$\overline{W}$	Write	12	I10	8X305 Instruction
13	—	No Connection	14	I9	8X305 Instruction
15	—	No Connection	16	I8	8X305 Instruction
17	—	No Connection	18	I7	8X305 Instruction
19	—	No Connection	20	I6	8X305 Instruction
21	CE1	Chip Enable	22	I5	8X305 Instruction
23	A0	8X305 Address (MSB)	24	I4	8X305 Instruction
25	A1	8X305 Address	26	I3	8X305 Instruction
27	A2	8X305 Address	28	I2	8X305 Instruction
29	A3	8X305 Address	30	I1	8X305 Instruction
31	A4	8X305 Address	32	I0	8X305 Instruction (MSB)
33	A5	8X305 Address	34	E7	Extended Microcode (LSB)
35	A6	8X305 Address	36	E6	Extended Microcode
37	A7	8X305 Address	38	E5	Extended Microcode
39	A8	8X305 Address	40	E4	Extended Microcode
41	A9	8X305 Address	42	E3	Extended Microcode
43	A10	8X305 Address	44	E2	Extended Microcode
45	A11	8X305 Address	46	E1	Extended Microcode
47	A12	8X305 Address (LSB)	48	E0	Extended Microcode (MSB)
49	V_{CC}	+5 V	50	V_{CC}	+5 V

THEORY OF OPERATION

As can be noted with the aid of the block diagram in Figure 5-1, the 8X305 prototyping system board contains circuits which may be categorized as follows:

1. Monitor Processor
2. 8X305 MicroController and Family
3. Writeable Control Store
4. Run/Step Logic

5.1 FUNCTIONS OF THE MONITOR PROCESSOR

The Monitor Processor, an 8035 microprocessor and peripherals, controls all the commands and operations described in Chapter 3. The Monitor Processor handles all communication with the terminal as well as reading and writing the 8X305's program storage, registers, and I/O port contents.

The 8X305 can execute instructions from Writeable Control Storage or an instruction that is latched into the 8243 by the Monitor Processor. Typically the instruction in the 8243 will store or read an 8X305's register contents, I/O Port contents, or set the address in the 8X305 Program Counter.

The 8243 is also used to read and write the contents of Writeable Control Store. Since the 8X305 does not have an address bus that can be three-stated and because a buffer would increase the memory access time, to read a specific memory location a JMP is "forced" upon the 8X305 by way of the 8243 to set the address lines.

The Monitor Processor reads the register contents of the 8X305 by forcing an XEC Rn, 000, where Rn is the desired register. This causes the register contents to be placed on the lower eight address lines of the 8X305 where it may be read by the Monitor Processor and sent out on the RS-232 interface. To store a value into the 8X305, the Monitor Processor will force a XMT Rn, XXX, where XXX is the desired register contents. (For R12 and R13, this will be accomplished by a XMT followed by a MOV.)

5.2 8X305 FAMILY

With the following two exceptions the 8X305 MicroController and its supporting peripherals connect to the prototyping system in a conventional manner:

1. Rather than a direct tie to the MCLK output of the 8X305, the MCLK input to the 8X310 Interrupt Control Coprocessor is gated. The gating circuits are required to implement correct single-step operation of the system.
2. The $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ inputs to the 8X305 are gated. Connected in this manner, the $\overline{\text{HALT}}$ and $\overline{\text{RESET}}$ signals will only affect the MicroController in the run mode. User circuits requiring either or both of these inputs should pick up the signals via the IV Bus connector J2.

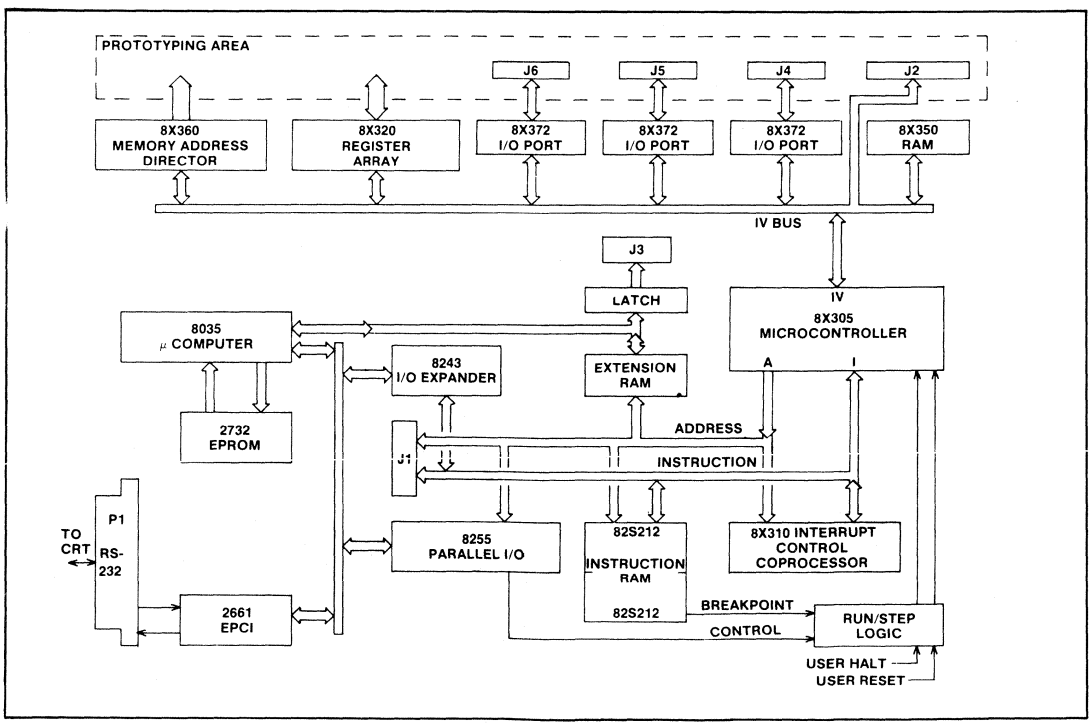


Figure 5-1. Detailed Block Diagram

5.3 WRITEABLE CONTROL STORAGE

Instead of the usual PROM or ROM instruction storage found in a typical 8X305 based system, a Writeable Control Store (WCS) has been implemented with high speed RAM to facilitate programming via the RS-232 terminal. The RAM memory provides 256 x 16 bits for 8X305 instruction storage, 256 x 8 bits for extended microcode, and 256 x 1 bit for breakpoints. If extended microcode is not desired the RAM chip at U21 may be removed and references to the extension will be removed from the display. Any one or all memory address locations may contain a breakpoint.

Note that no page decoding is provided on the board, so the 256 words of instructions will be repeated every 256 addresses throughout the entire 8K memory range of the 8X305.

The memory may be expanded up to the full 8K directly addressable by the 8X305. A 4K word Writeable Control Storage Expansion Module is available from Signetics (Part Number 8X300KT2SK). When the additional memory is installed in J1, the RAM enable (RAMEN) signal is grounded to disable the on-board 256-word memory, and the Monitor Processor is signaled to provide the correct write cycle at J1 for the added RAM. See Figure 5-2 for the differences:

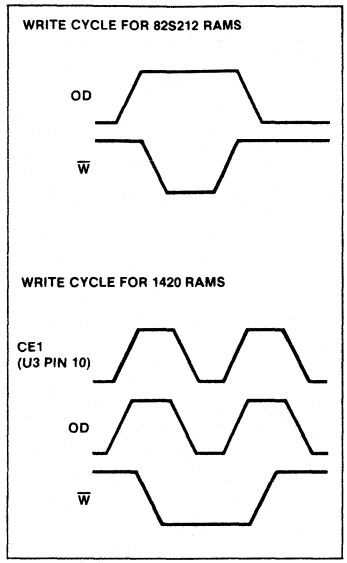


Figure 5-2. Write Cycle Variations

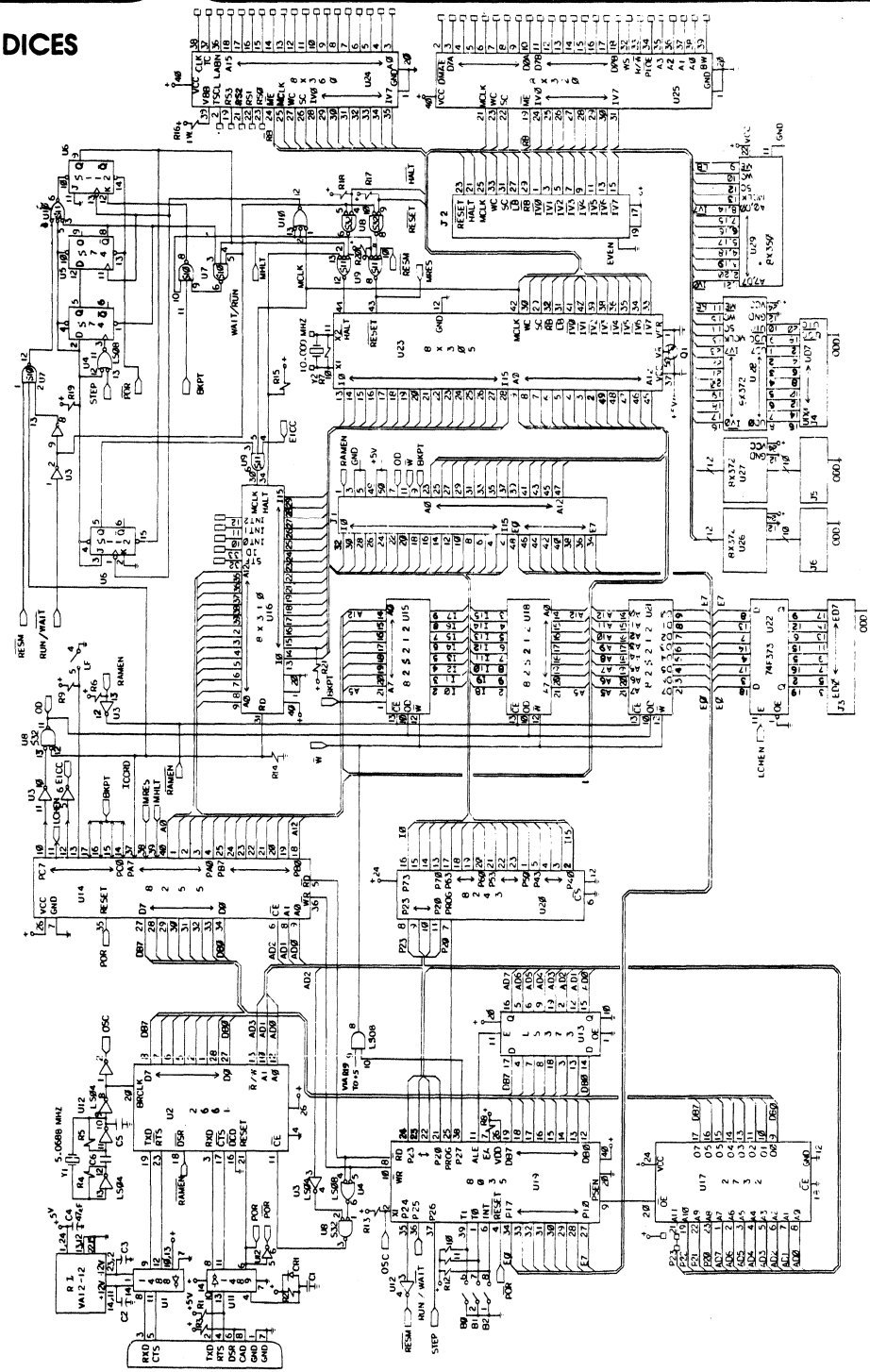
5.4 RUN STEP LOGIC

The Run-Step logic consists of components U5, U6, U7, U9 and U10 on the schematic in Appendix A. These circuits provide the control logic required to allow the 8X305 to execute instructions at full speed or in a single step mode of operation. This is easily accomplished since all instructions are executed within one machine cycle, the time from the falling edge of MCLK to the next falling edge of MCLK. The $\overline{\text{HALT}}$ input is sampled by the 8X305 sometime after the falling edge of MCLK. If it is low, the address lines of the MicroController are held stable; the current instruction is executed after the $\overline{\text{HALT}}$ input goes high (inactive). During the time that the $\overline{\text{HALT}}$ input is low (active), the MCLK output is unaffected. Inputs to the Run-Step logic are labeled RUN/$\overline{\text{WAIT}}$, STEP, BKPT and MCLK; the output is labeled $\overline{\text{HALT}}$ and connected directly to the $\overline{\text{HALT}}$ input of the 8X305 MicroController.

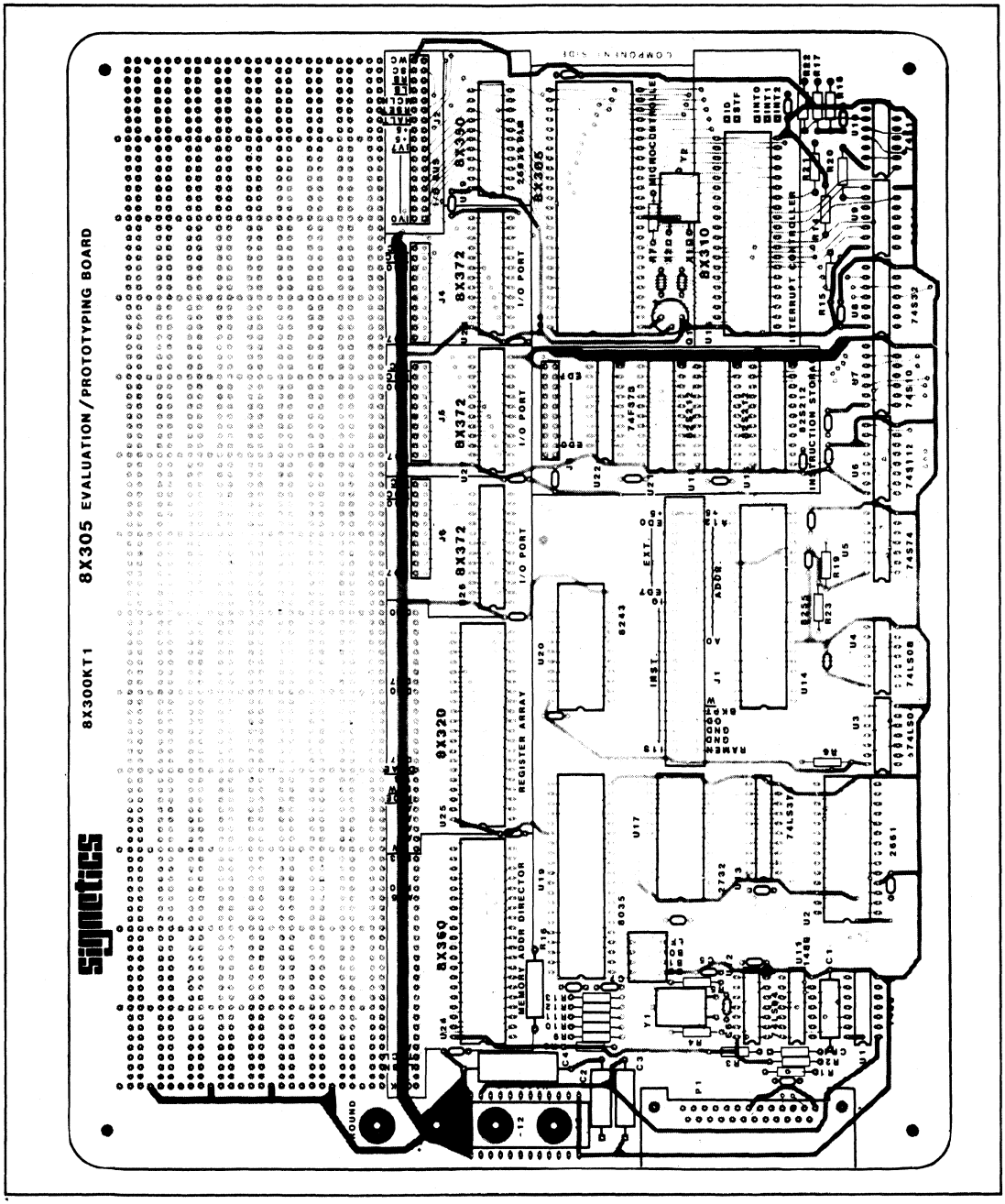
Two of the inputs, RUN/$\overline{\text{WAIT}}$ and STEP, are controlled by the Monitor Processor. The BKPT input connects to the extra bit in WCS that is used for breakpoints. The MCLK input comes directly from the MCLK output of the 8X305.

During single stepping the RUN/$\overline{\text{WAIT}}$ line is low and a pulse on the STEP line causes the 8X305 to execute only the current instruction. This is because the $\overline{\text{HALT}}$ line will go high for just one machine cycle. Entering the run mode the RUN/$\overline{\text{WAIT}}$ line is high and a pulse on the STEP line causes the 8X305 to begin executing instructions from the current address at full speed. The $\overline{\text{HALT}}$ input will go high and remain so until the RUN/$\overline{\text{WAIT}}$ line is brought low or until a breakpoint is encountered.

APPENDICES



Appendix A. 8X300KT1SK Prototyping System Schematic



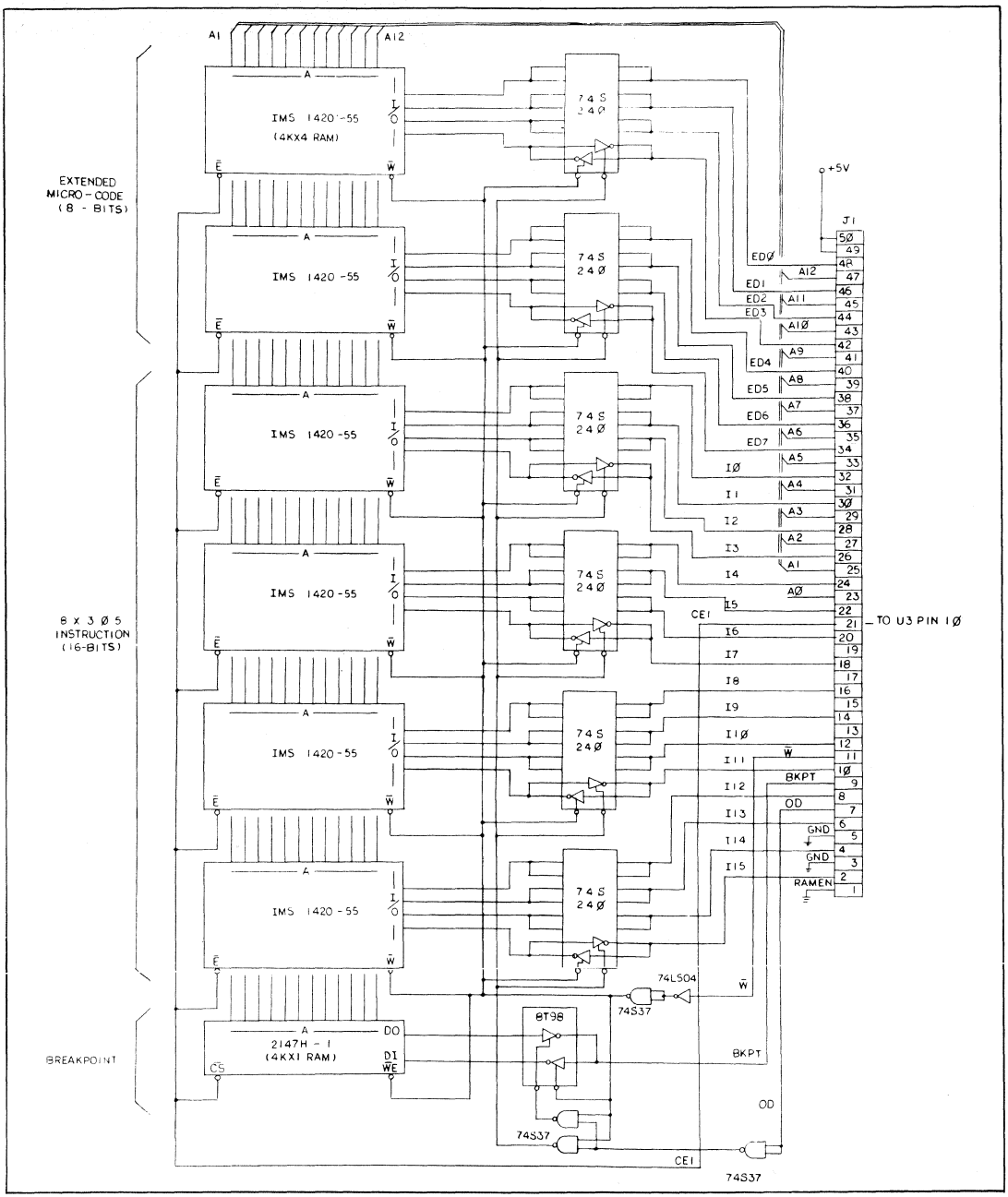
Appendix B. 8X300KT1SK PCB Layout and Parts Placement

Item No.	Manufacturer	Part Number	Description	Designator	Qty.
1	Signetics	MC1488N	Quad Line Driver	U1	1
2	Signetics	SCN2661CC1N28	EPCI	U2	1
3	Signetics	N74LS04N	Hex Inverter	U3, U12	2
4	Signetics	N74LS08N	Quad AND Gate	U4	1
5	Signetics	N74S74N	Dual D-Flip-Flop	U5	1
6	Signetics	N74S112N	Dual J-K Flip-Flop	U6	1
7	Signetics	N74S10N	Triple NAND Gate	U7	1
8	Signetics	N74S32N	Quad OR Gate	U8	1
9	Signetics	N74S11N	Triple AND Gate	U9, U10	2
10	Signetics	MC1489N	Quad Line Receiver	U11	1
11	Signetics	N74LS373N	Octal Latch	U13	1
12	Intel	P8255A	PPI	U14	1
13	Signetics	82S212N	256 x 9 RAM	U15, U18, U21	3
14	Signetics	N8X310N	Interrupt Control Coprocessor	U16	1
15	Intel	2732A-4	4096 x 8 EPROM	U17	1
16	Signetics	SCN8035AC6N40	8-Bit Microcomputer	U19	1
17	Intel	P8243	Input/Output Expander	U20	1
18	Signetics	N74F373N	Octal Latch	U22	1
19	Signetics	8X305I/N	MicroController	U23	1
20	Signetics	N8X360N	Memory Address Director	U24	1
21	Signetics	N8X320N	Register Array	U25	1
22	Signetics	N8X372-000N	I/O Port	U26	1
23	Signetics	N8X372-001N	I/O Port	U27	1
24	Signetics	N8X372-002N	I/O Port	U28	1
25	Signetics	N8X350N	256 x 8 RAM	U29	1
26	ITT CANNON	DBP-25SAA	RS-232 Connector	P1	1
27	TRW CINCH	252-25-30-360	Edge Connector, 50 Pin	J1	
28	Spectra-Strip	800-579	Header, 34 Pin	J2	1
29	Spectra-Strip	800-586	Header, 20 Pin	J3, J4, J5, J6	4

Appendix C. Parts List

Item No.	Manufacturer	Part Number	Description	Designator	Qty.
30	Reliability	VA12-12	DC-DC Converter		
31	Saronix	NMP051L	Crystal, 5.688 MHz	Y1	1
32	Saronix	NMP100L	Crystal, 10.000 MHz	Y2	1
33		2N5320	Transistor	Q1	1
34	ALCO	DSS-4	Switch, Mini-dip	B2, B1, B0, LF	1
35	Smith	230	Binding Post		4
36		1N914	Diode	CR1	1
37			CAP, 0.1 μF		28
38			CAP, 47 μF, 20 V	C2, C3, C4	3
39			CAP, 47 μF, 6 V	C1	1
40			CAP, 47 pF	C5	1
41			CAP, 0.1 μF	C6	1
42			RES., 1K, 1/4 W	R19	1
43			RES., 10K, 1/4 W	R1, R2, R3	3
44			RES., 390, 1/4 W	R4, R5	2
45			RES., 2.2K, 1/4 W	R6-R13, R15, R17, R18, R20, R21	13
46			RES., 18, 1 W	R16	1
47	H.H. Smith	2501	Bolt, Nylon 4 — 40 x 3/8"	P1	2
48	H.H. Smith	2554	HEX Nut, Nylon 4 — 40	P1	2
49	BURNDY	DILBQ50P-101	Socket, 50 Pin	U23	1
50	T.I.	C844002	Socket, 40 Pin	U14, U16, U19, U24, U25	5
51	T.I.	C842802	Socket, 0.6" 28 Pin	U2	1
52	T.I.	C842402	Socket, 0.6" 24 Pin	U17, U20	2
53	EMC	17424-01-445	Socket, 0.4" 24 Pin	U26, U27, U28	3
54	T.I.	C842202	Socket, 0.4" 22 Pin	U15, U18, U21, U29	4
55	T.I.	C842002	Socket, 0.3" 20 Pin	U22	1
56	Signetics	PCB-82001	P.C. Board		1
57	H.H. Smith	2450	Rubber Bumper		5

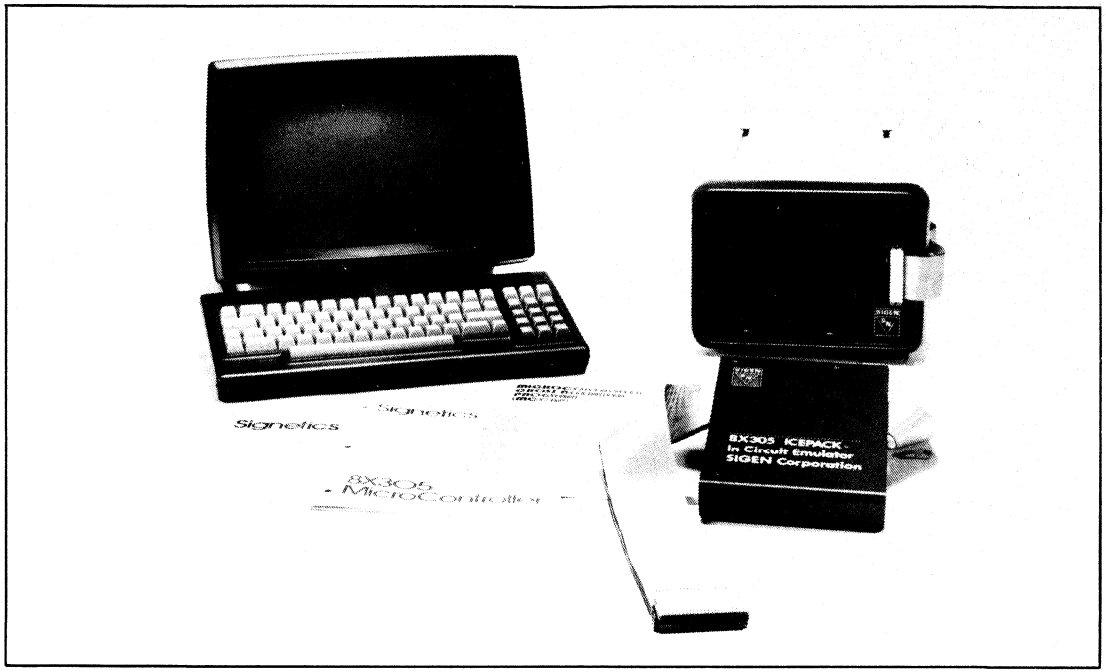
Appendix C. Parts List continued



Appendix D. Memory Expansion Assembly Schematic

ICEPACK

Originally published by Signetics January 1984



FEATURES

- Diagnostic Monitor for controlled program execution, 32 breakpoints, register, memory and I/O port examination/change, download/upload memory
- In-line assembler/disassembler for fast debugging of 8X305 code in symbolic assembly language
- System trace memory for 128 addresses, 12 bites each
- On-board emulation memory of 4096 × 24 bits
- Supports all other 8X305 family devices
- Supports microcoded designs using expanded instruction widths
- Power-on diagnostics
- Emulator and software runs with other CP/M* or ISIS* based systems

COMPLETE HARDWARE/SOFTWARE DEVELOPMENT SYSTEM INCLUDING:

- 8X305 Emulator module
- CP/M 2.2 System — Z80A, 64K RAM, dual minifloppies with 1.6 Mbytes storage
- Screen oriented editor for easy program development
- Cross Assembler supporting Signetics standard format mnemonics
- IBM P.C. version available

FASTER PRODUCT DEVELOPMENT

Demanding control applications based on the low cost, high performance Signetics 8X305 bipolar microcontroller can now be developed and implemented quickly and economically with the SIGEN 8X305 ICEPACK.

ICEPACK is a powerful, high performance development and in-circuit emulation system for use with the 8X305 series microcontroller product family. Designed for CP/M compatibility, ICEPACK provides a cost-effective means of rapid product development without the costly dedicated resources previously required.

---

*CP/M is a trademark of Digital Research Corp.
*ISIS is a trademark of Intel Corporation.

PERFORMANCE YOU NEED

ICEPACK provides both the hardware performance and software tools needed for efficient 8X305 product development throughout a broad range of applications. Designed specifically by SIGEN for the development of 8X305 based products, ICEPACK performance has been proven throughout a variety of products. Its capabilities are especially useful in real time control applications.

The ICEPACK Emulator module simply plugs into the prototype system 8X305 socket through a flat ribbon cable. The ICEPACK's superior noise immunity eliminates typical problems associated with circuit interfacing. The ICEPACK Emulator hardware features high speed electronics supporting clock operation up to 10 MHz. Rugged packaging assures long life and reliable operation.

ICEPACK software includes:

- Powerful screen oriented Editor for easy program devlopment.
- Full featured Cross Assembler supporting standard Signetics mnemonics.
- Powerful diagnostic Monitor enabling controlled program execution including single stepping, breakpoint setting, memory and register examination an modification.
- In-line assembler/disassembler for easy assembly language and debug and patching.

FITS YOUR REQUIREMENTS

ICEPACK is available ready to use as a system, or ready to interface to your floppy based CP/M system. Either way, you get the same powerful ICEPACK System capabilities. The ICEPACK System includes a Z80A based, 64K RAM, dual floppy system, ICEPACK Emulator, interface adapter, CP/M 2.2 and all ICEPACK software and manuals. The ICEPACK Subsystem includes Emulator, parallel interface adapter, interface cables, ICEPACK software, and user manuals.

TECHNICAL SPECIFICATIONS

Power Requirements: 115/230VAC, 100 W

Physical	(CP/M System)	(Emulator)
Height	7.5 inches 190mm	1 inch 25mm
Width	9.5 inches 241mm	6 inches 152mm
Depth	14.5 inches 368mm	8 inches 204mm
Weight	13.0 pounds 5.9 kg	1 pound .45 kg

Environmental

Operating Temperature: 0°C to +40°C
Storage: −40°C to +85°C

Cables

(Target System)

— 100 wire flat cable, 18 inches long
— 20 wire flat cable, 18 inches long

(CP/M System)

— 37 wire flat cable, 5 feet long

Enclosure

(Emulator)

— Anodized brushed aluminum

(CP/M System)

— High impact plastic with internal shielding

DEVELOPMENT DATA I/O PORT PROGRAMMER

Originally published by Signetics January 1984

Hardware Features

- Real time in-circuit emulation to 10 MHz without wait states.
- 8K words of 35ns RAM, 16 or 32-bit words.
- Memory mapping in 1K word increments.
- Compatible with 8X310 Interrupt Control Chip.
- Trace capability includes 128 cycles of address, IV bus, RB, LB, WC, SC, and 3 selectable points in target system. Both input and output phases are traced in each cycle.
- Downloading and Uploading capability provided.
- I/O Port Programmer for 8X372, 8X376, 8X382.

Software Features

- Relocating macroassembler compatible with Signetics MCCAP.
- Linking editor permits linking separately-assembled modules to form one load module.
- Debugger program controls single stepping, stopping on a specified address, printing trace information, disassembling, program patching, changing register contents and memory management. Symbolic debug capability is provided.
- Command file capability provided in the Asembler, Linking Editor, and Debugger program.
- PROM formatting program available. Slices 16 and 32-bit words into 4 or 8-bit groups.

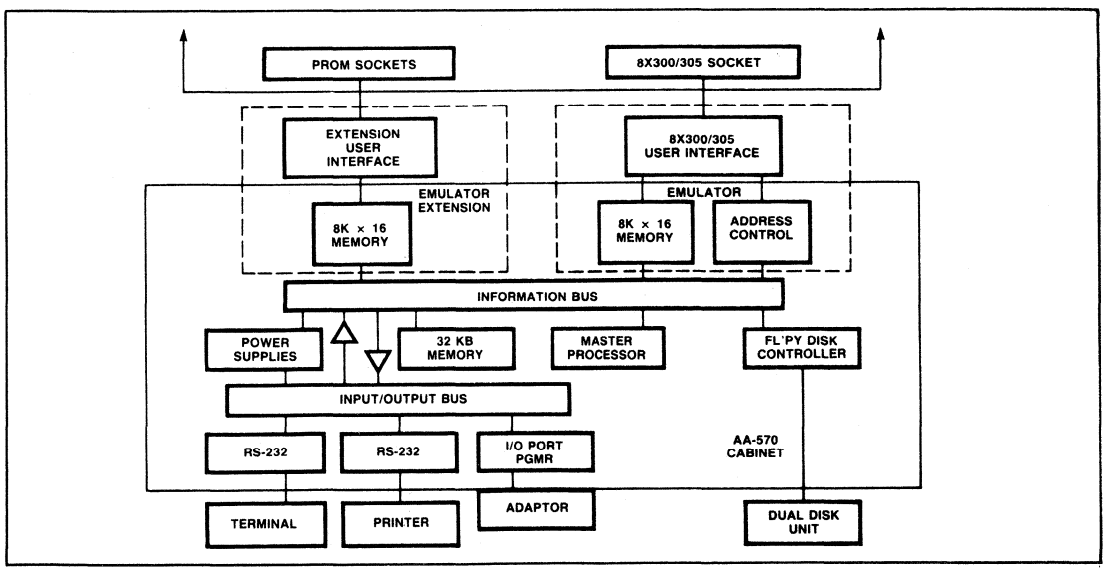
BLOCK DIAGRAM

The block diagram shows how the 8X300/305 support devices are incorporated in an EZ-PRO system. Devices unique to the 8X300/305 includes the AA-572-8X35 In-Circuit Emulator, the AA-572-8X35-M Emulator Extension and the AA-574-8X37 I/O Port Programmer. The Emulator Extension provides an extra sixteen bits of word length over the basic sixteen bits required for the 8X300/305 processor and may or may not be required in a particular development. Note that the emulator consists of three printed circuit board assemblies and the extension, two.

The Address Control assembly incorporates trace memory and logic for memory mapping, stopping, and single stepping as well as circuitry required to communicate with the User Interface and Master Processor. Each Emulator Memory assembly is equipped with 8K 16-bit words of 35 ns memory as well as interface circuitry. This memory is loaded and unloaded under control of the Master Processor and is accessed by addresses generated by the 8X300/305.

The 8X300/305 User Interface has the processor mounted on it along with cable termination networks, cable drivers, some logic and test points. Test points are provided for the three points which may be traced in the target system, connection to the 8X310 Interrupt Control Chip and oscilloscope sync.

TARGET SYSTEM



BLOCK DIAGRAM OF EZ-PRO CONFIGURED TO SUPPORT 8X300/305 DEVELOPMENT WITH 32-BIT WORK LENGTH

The Extension User Interface is equipped with six DIP sockets and cables which permit connection into PROM sockets located in the target system. Four of the sockets are 18 pin (for 4-bit wide PROMs) and two are 24 pin (for 8-bit wide PROMs). Either set of sockets and cables may be used. Pin outs are compatible with Signetics 82S137, 185, 181, and 191 PROM types.

PROM programming is supported in two ways. Both require that PMFORM, the PROM formatting program, be utilized. After PMFORM has created files consisting of either 4-bit or 8-bit wide slices of the object program words, these files may be directed to a DATA I/O (or equivalent) PROM programmer connected to the RS232 printer port. Alternatively, with 8-bit wide slices, the AA-574-27XX EPROM Programmer may be utilized to write the files into 2716 or 2732 EPROMs. With appropriate off-line equipment, information in the EPROMs may be transferred into bipolar PROMs.

The I/O Port Programmer consists of a printed board assembly, an adaptor that fits into the ZIF socket located on the front of the AA-570 Basic Development Unit and a program. After checking to see that the 8X372/376,382 is properly oriented in the ZIF socket and fuses are unblown, the program permits the desired address to be programmed into the I/O port. Complete checking is then accomplished including validation of the address and transfer of information in both directions through the port.

EZ-PRO SYSTEM ELEMENTS FOR 8X300/305 SUPPORT

Model	Description
AA-570-200	Basic Development Unit
AA-59X	Dual Disk Unit
AA-572-8X35	In-Circuit Emulator for 8X300/305
AA-572-8X35-M	Emulator Extension
AA-562	Printer, RS232 Interface
AA-563	Video Terminal with Screen Editor
AA-553	PMFORM, PROM Formatter Program
AA-574-8X37	I/O Port Programmer
AA-574-27XX	EPROM Programmer for 2716 & 2732

Note that all required programs except PMFROM are supplied at no extra cost.

303A-8X PROGRAMMING TEST ADAPTER

The 303A-8X Programming Test Adapter is designed to program address fuses and activate protect fuses for Signetics' I/O Ports 8X372, 8X374, 8X376, and 8X382. Error messages are displayed if the programmed part is defective or if ambiguous addresses occur during the programming procedure. The test adapter operates in conjunction with the Data I/O Logic PAK 303A-V01 and various models of the Data I/O (System 19, 29A, and 100A). The Programming Test Adapter is quick and easy to use and most Signetics' Franchised Distributors provide on-site programming capabilities for customer parts.

ECC APPLICATION NOTE

Originally published by Signetics January 1984

INTRODUCTION

The widespread use of floppy/hard disk technologies in Commercial and Military environments has created the need for high-speed disk controllers that are capable of both error-detection and error-correction capabilities. At the present time, most disk controllers are limited to error-detection only, with one of the more widely used schemes being that of a Cyclic Redundancy generator Checker (CRC) and associated software/hardware support. Although the CRC technique is virtually foolproof from an error-detection point of view, it has no provision for error correction. With bit and track densities constantly on the increase, it is desirable to implement a reliable error-correction feature to improve the performance and reliability of the disk controller.

This application note describes a software approach to error correction for the Signetics 8X305/8X330 based floppy disk controller and 8X305 based hard disk controller. A typical floppy disk controller system is shown in Figure 1. This error correction algorithm can correct up to a 9-bit single burst error and is fully compatible with controllers that use the CRC-CCITT ($X^{16} + X^{12} + X^5 + 1$) polynomial. The Signetics 8X330 floppy disk controller chip provides some unique features, unmatched by other LSI chips, which allow the 8X305 MicroController firmware to implement the error correction procedure.

GENERAL SCHEME FOR ERROR DETECTION AND CORRECTION

Error Detection by CRC Polynomial. The CRC scheme for error detection is implemented with a CRC generator. During transmission, a block of binary serial data passes through a preset CRC generator to generate the necessary and unique check bits for a particular block of data. The check bits are appended to the end of the data block. The complete data and check bits are then transmitted to the receiver. At the receiver, the complete data and check bits pass through the same preset CRC generator. If no errors occurred during data transmission, the CRC generator must have a remainder of all zeros.

Hardware implementation of the CRC generator is simply a feedback shift register with EX-OR gating. Figure 2 shows the equivalent circuit for the CRC-CCITT ($X^{16} + X^{12} + X^5 + 1$) generator. The CRC-CCITT polynomial is the industry standard used in floppy disk controllers; however, if desired, other CRC polynomials can be easily implemented. In any case, there are as many register stages as the highest order of the polynomial and the number of EX-OR gates is one less than the number of X-terms. The detection span for a particular polynomial equals the number of the highest order X-term. For CRC-CCITT, the detection span is 16 bits.

Error Correction by Reverse CRC Polynomial. Several different error correction algorithms are currently used in high-performance disk systems. The error correction procedure described in this application note is based on the reciprocal polynomial algorithm.

The reciprocal or reverse polynomial for a particular forward polynomial can be derived by subtracting the order of every X-term of the forward polynomial from the highest order of the forward polynomial. For example, the CRC-CCITT polynomial is $X^{16} + X^{12} + X^5 + 1$, hence the CRC-CCITT reverse polynomial is $X^{16} + X^{11} + X^4 + 1$ (i.e., $X^{16-16} + X^{16-12} + X^{16-5} + X^{16-0}$). Figure 3 shows the equivalent circuit and the hardware implementation for the CRC-CCITT forward/reverse polynomial.

As described previously, error detection is achieved by examining the remainders of the CRC generation after the complete data and CRC check bits have passed through the CRC generator. A nonzero remainder indicates a read-data error. A nonzero remainder contains information to determine the error pattern and its location. Error correction is achieved by manipulating the nonzero remainder in a reverse CRC generator.

The error correction procedure begins with the loading of the nonzero remainder into the reverse CRC generator in reverse order. The Most Significant and Least Significant Bits

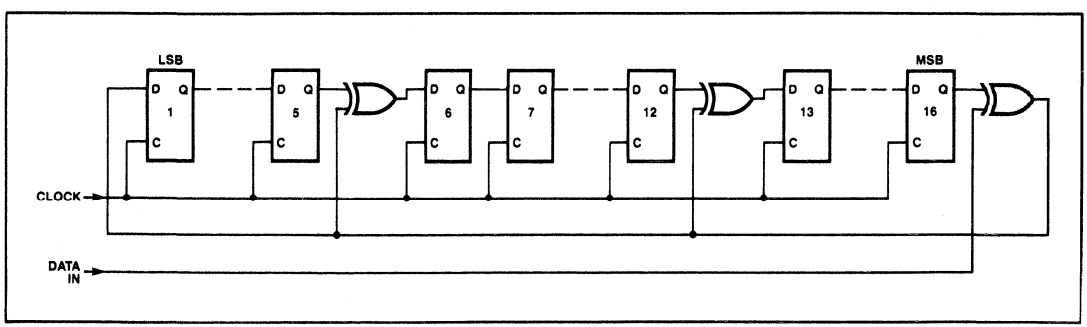
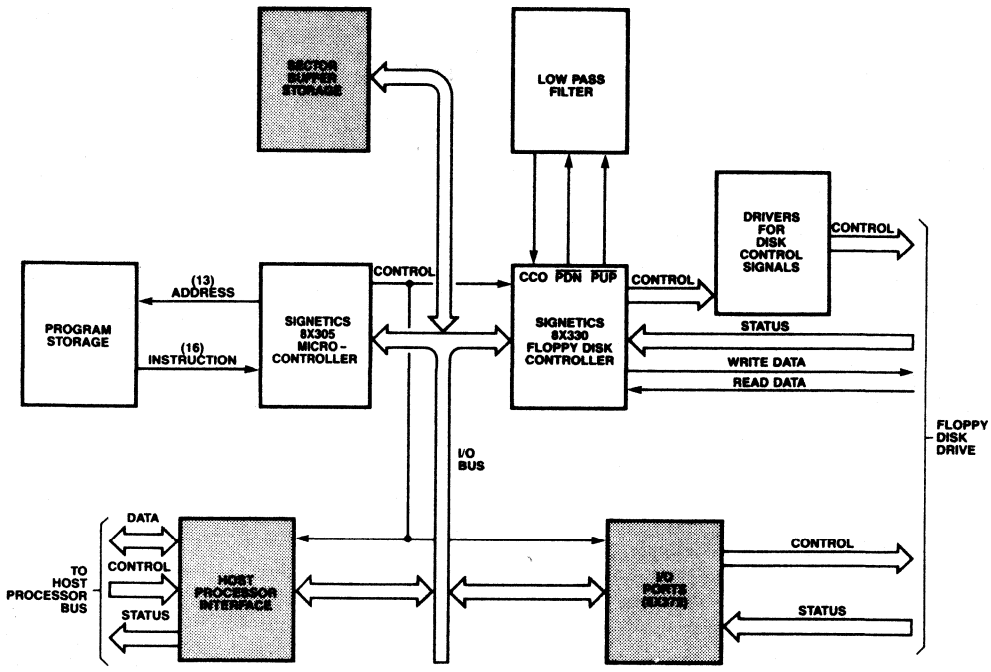


Figure 2. Equivalent Circuit for CRC-CCITT ($x^{16} + x^{12} + x^5 + 1$)



Note:
1. Shaded blocks may change to suit users system requirements.

Figure 1. 8X330 Based Floppy Disk Controller-Block Diagram

of the remainder are loaded into the Least Significant and Most Significant Bits of the reverse CRC generator, respectively. The DATA IN input of the reverse CRC generator is connected to ground to provide zero data input during the complete error correction process. Clocking is then provided for the CLOCK input. The contents of the generator are examined for an error condition after each clock cycle. If the generator has "n" register stages and "m" is the desired single burst correction span, then the error condition is characterized by (n-m) consecutive zeros at specified bit locations. The number of clock cycles needed to find the error pattern provides the error location information.

ERROR DETECTION AND CORRECTION SCHEME FOR 8X305/8X330 BASED CONTROLLER

Features of 8X305/8X330 Based Floppy Disk Controller. The following error correction scheme for 8X305/8X330 based disk controllers requires two forward CRC generators (CRC1F and CRC2F) and two reverse CRC generators (CRC1R and CRC2R). The CRC1F generator has been implemented in the 8X330 hardware. The CRC2F, CRC1R, and CRC2R generators are implemented in the 8X305 firmware. One unique feature of the 8X330 is that the remainder in the internal CRC generator can be read by the controlling processor. This feature provides the required information for the 8X305 MicroController to do error correction in the firmware. The MOS LSI floppy disk controller implementation does not provide this flexibility and only generates a read error status flag for error detection.

The speed of the 8X305 MicroController and its bit-manipulation oriented instruction set, combined with the 8-bit parallel CRC calculation algorithm described below allows the 8X305 to compute CRC for 8 bits of data input in 4 to 5 microseconds. Thus, even with a data transfer rate of 500K bits/sec for a double density 8-inch floppy disk drive, the 8X305 is fast enough to control the 8X330 and compute the CRC for CRC2F in software on-line, as each data byte is transferred between the 8X305 and 8X330 during the read and write cycles.

Two Forward CRC Polynomials. The first 16-bit forward CRC polynomial (CRC1F) is the CRC-CCITT ($X^{16} + X^{12} + X^5 + 1$), which is the standard polynomial used in floppy disk controllers. The second 16-bit forward CRC polynomial (CRC2F) is $X^{16} + X^{13} + X^{12} + X^{10} + X^8 + X^6 + X^5 + X^4 + 1$. This polynomial is chosen by computer search to achieve a single burst correction span of 9 bits. A 16-bit CRC check word is generated for each of the two polynomials. These two 16-bit CRC check words are appended to the end of the transmit data. The CRC-CCITT is chosen for the first polynomial to achieve media compatibility with old (8X300)

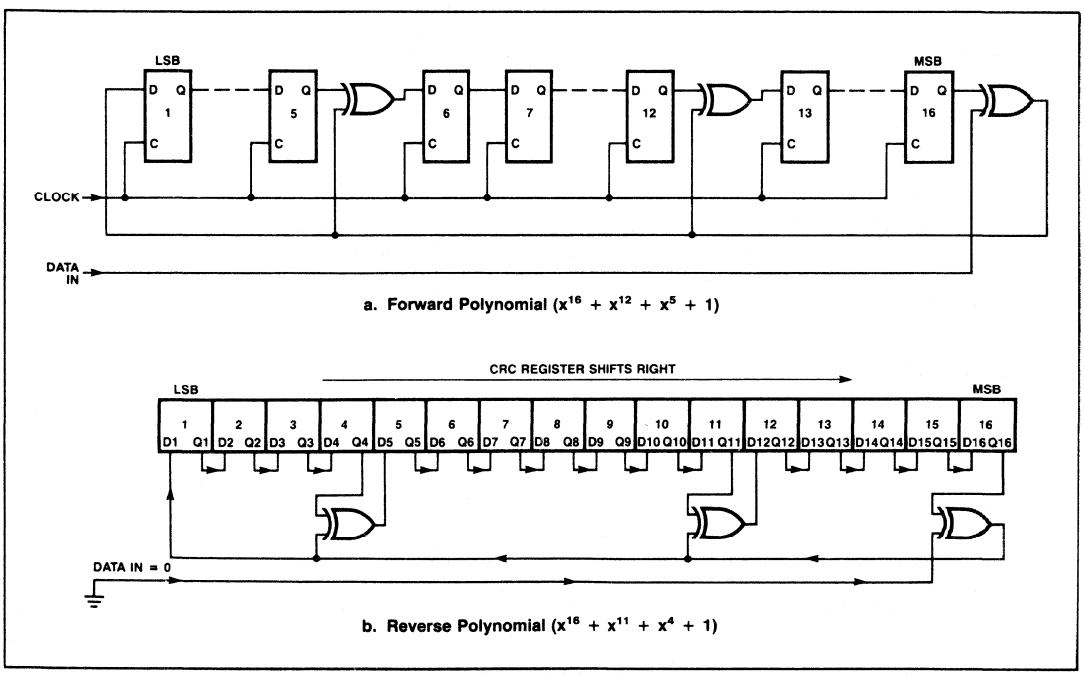


Figure 3. Equivalent Circuit and Hardware Implementation for CRC-CCITT Forward/Reverse Polynomial ($x^{16} + x^{11} + x^4 + 1$)

controllers without error correction capability and the 8X305/8X330 based controller implemented with the proposed error correction scheme. A disk generated by earlier 8X300 MicroControllers can be read by the controller with the proposed error correction scheme and vice versa.

Software Implementation of the CRC2F Generator. The hardware implementation of the CRC2F generator is illustrated in Figure 4. The CRC2F generator shifts left one bit each clock cycle. The most significant bit is located on the left side of the generator. Left shift is used to simplify the software implementation, because the MSB of the data byte transferred between the 8X305 and 8X330 is on the left side. The 8-bit parallel CRC generation mechanism is shown in Figure 5. The initial contents of the generator are represented by symbols "a b c . . . n o p" with "a" and "p" being the MSB and LSB, respectively.

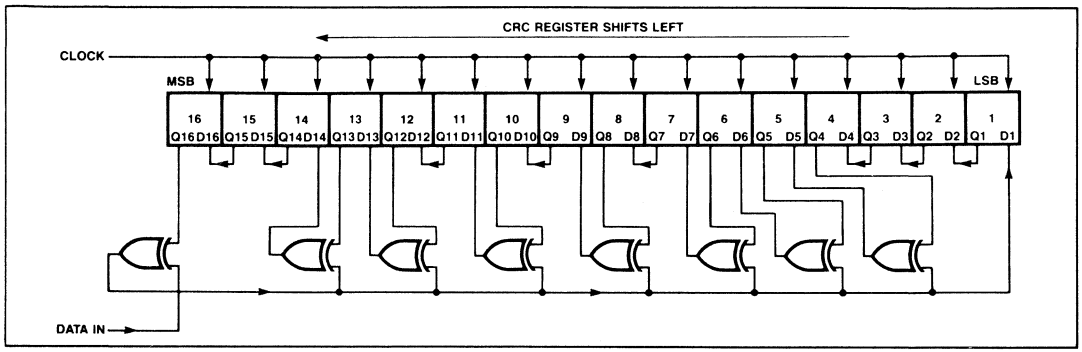


Figure 4. Hardware Implementation for $x^{16} + x^{13} + x^{12} + x^{10} + x^8 + x^6 + x^5 + x^4 + 1$ (CRC2F)

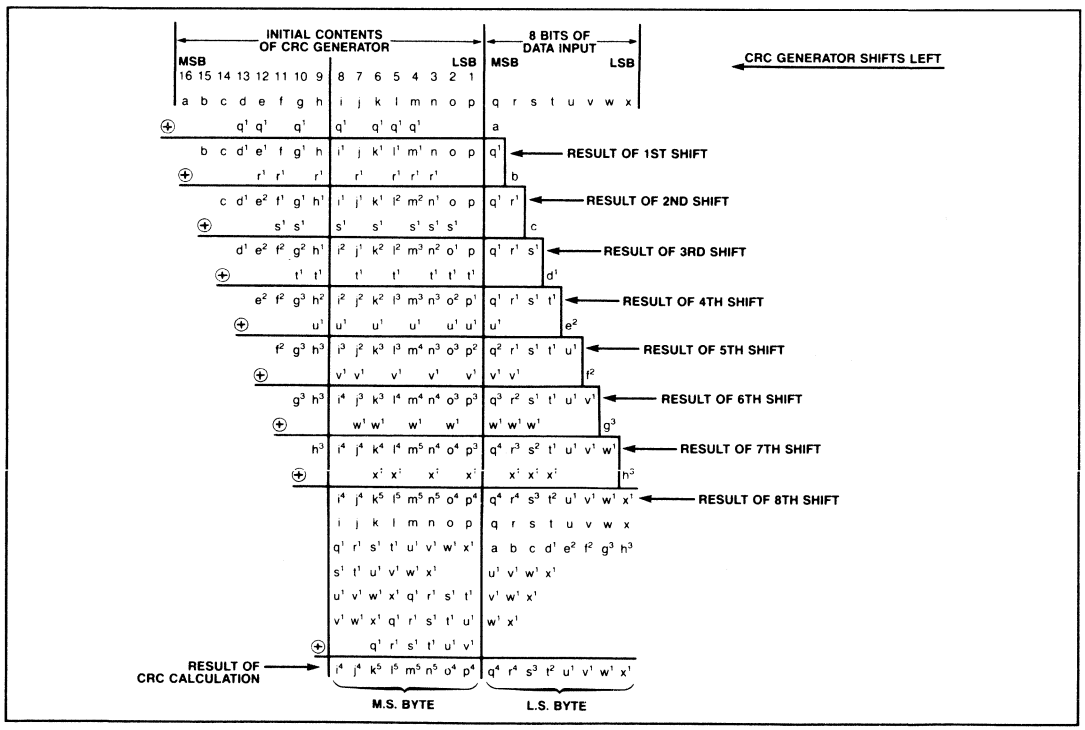


Figure 5. 8-Bit Parallel CRC Generation Mechanism for $x^{16} + x^{13} + x^{12} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^1$

The eight bits of data input are symbolized by "q r s t u v w x", with "q" being the MSB. The top part of Figure 5 is a symbolic representation of the eight left shift operations of the CRC generator shown in Figure 4, for the eight serial data input bits. As the first data bit "q" comes in, it is EX-ORed with the MSB of the CRC generator. The result of this operation, "q'", is then fed back and EX-ORed with the outputs of several register stages as shown in Figures 4 and 5. Then the generator is clocked and shifted left by one bit postion. This sequence of operations is repeated for the eight data inputs shown at the top of Figure 5. The bottom part of Figure 5 summarizes the number of EX-OR operations required to derive the result. The operands of the EX-OR operations are mostly the field, rotated field, or subfield of the byte "q'r's't'u'v'w'x". This special characteristic, together with the 8X305 bit manipulation instruction set, permits an 8-bit parallel CRC calculation for a 16-bit polynomial to be accomplished in 4 to 5 microseconds. Figure 6 shows the software steps required to implement the 8-bit parallel CRC calculation. The first step is to derive the byte "q'r's't'u'v'w'x'" from the 8-bit data input and the initial contents of the CRC generator by operations A, B, and C. The result is derived by a sequence of EX-OR operations as shown in Figure 6. The software implementation of CRC2F requires a total of 21 instructions, as shown in line numbers 448 to 471 of the program listing.

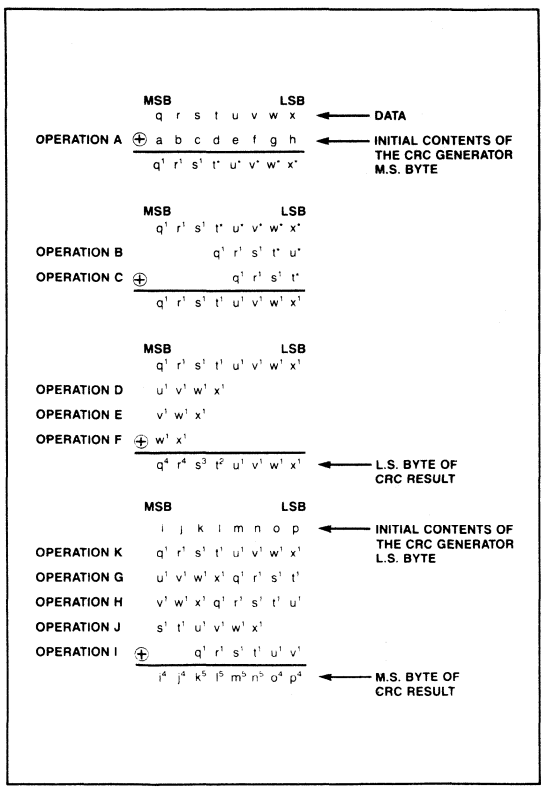


Figure 6. Steps Required for the Software Implementation of CRC2F

Two Reverse CRC Polynomials. Since the error correction scheme described in this application note uses the reciprocal polynomial algorithm, two reverse CRC generators, corresponding to CRC1F and CRC2F, are required during the correction cycle. The two reverse CRC generators are referred to as CRC1R and CRC2R.

$$CRC1R = X^{16} + X^{11} + X^4 + 1$$
$$CRC2R = X^{16} + X^{12} + X^{11} + X^8 + X^6 + X^4 + X^3 + 1$$

Software Implementation of the CRC1R and CRC2R Generators. The implementation of the CRC1R and CRC2R generators in software is similar to that of the CRC2F. There are two exceptions:

- During the correction cycle, the DATA IN inputs of the reverse CRC generators are connected to ground.
- The reverse CRC generators shift right, so that the MSB is on the right-hand side, thus simplifying the software. As errors are detected, the error correction procedure requires that the nonzero remainders of the forward CRC generators be loaded into the reverse CRC generators in reverse order; that is, the MSB from the forward CRC generator is loaded into the LSB of the corresponding reverse CRC generator. This process is achieved by loading the most significant byte of the forward generator into the least significant byte of the reverse generator, because the MSB of the forward generator is on the left side and the MSB of the reverse generator is on the right side.

Figures 7 and 8 show the mechanics and steps required to implement the CRC1R generator of Figure 3 in software. A similar procedure is used to implement the CRC2R generator. The software coding of the CRC1R and CRC2R are shown in the line numbers 487 through 541 of the program listing.

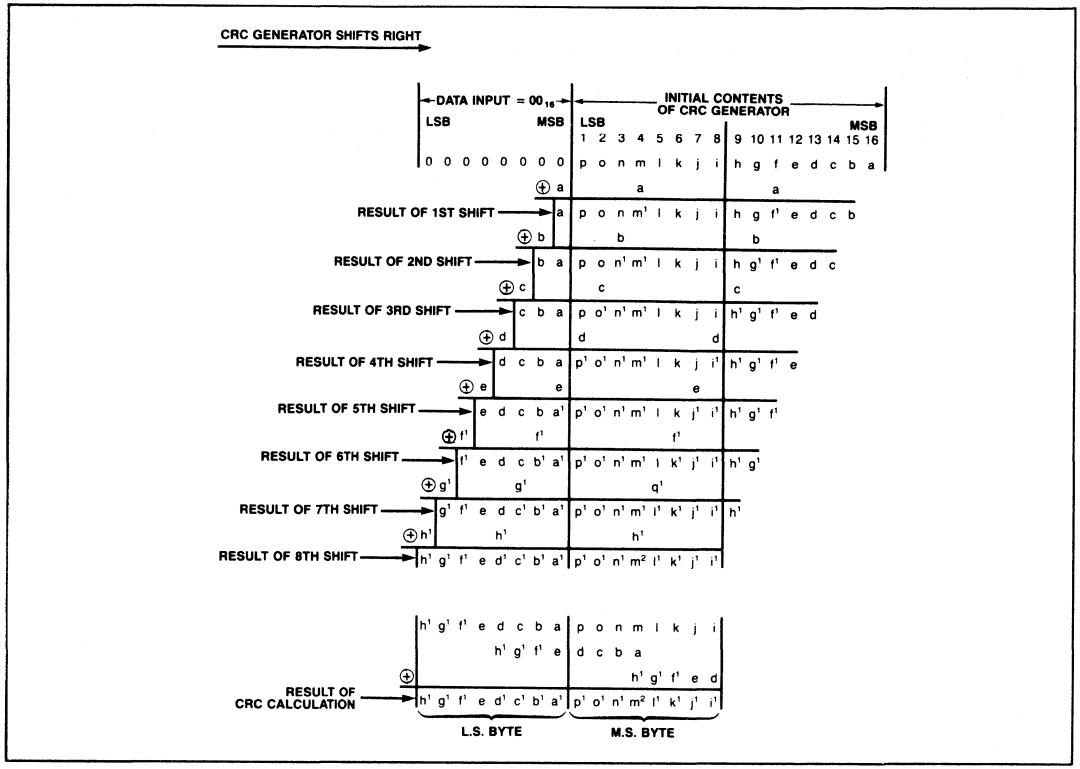


Figure 7. 8-Bit Parallel CRC Generation Mechanism for CRC1R

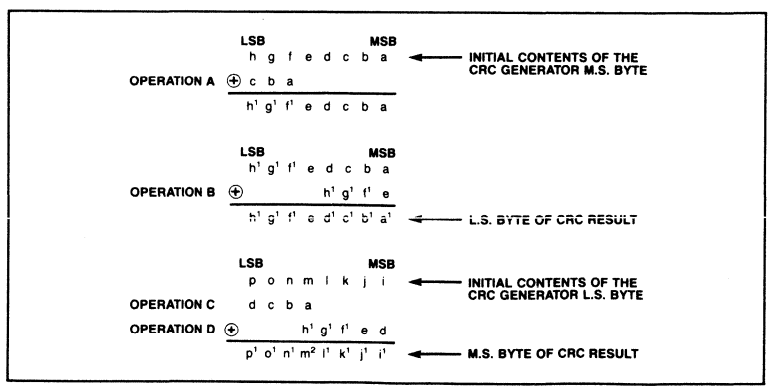


Figure 8. Steps Required for the Software Implementation of CRC1R

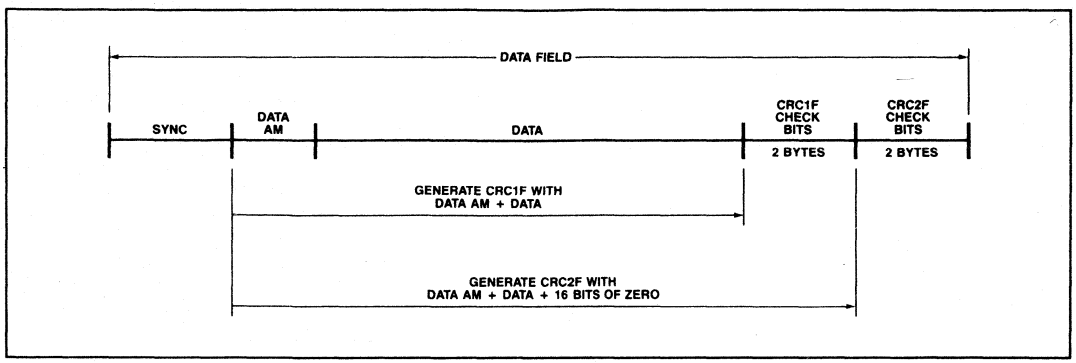


Figure 9. Write Cycle

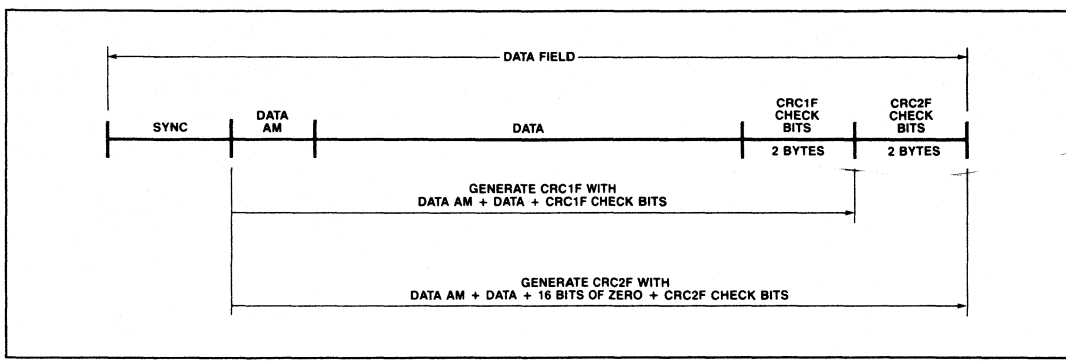


Figure 10. Read Cycle

Error Detection Scheme. As shown in Figure 9, four CRC check bytes are appended to the last bit of data during the write cycle. The first two check bytes are derived by passing the data address mark and data bytes through the preset CRC1F generator. After accepting the data address mark and data bytes, the contents of the CRC1F generator are the two CRC1F check bytes. The two CRC2F check bytes are generated by applying the data address mark, the data, and sixteen bits of zero as inputs to the preset CRC2F generator.

During the read cycle, the preset CRC1F generator calculates from the first bit of data address mark to the last bit of CRC1F check bit as illustrated in Figure 10. During the read cycle the data address mark, the data, sixteen bits of zero, and two CRC2F check bytes are input to the preset CRC2F generator. No error is detected if the remainders of the CRC1F and CRC2F generators are zeros. The types of errors associated with nonzero remainders are as follows:

CRC1F Remainder	CRC2F Remainder	
Zero	Zero	No error
Zero	Nonzero	Correctable error in CRC2F check bytes (or uncorrectable error)
Nonzero	Zero	Correctable error in CRC1F check bytes (or uncorrectable error)
Nonzero	Nonzero	Correctable error in data bytes

Error Correction Scheme. Once an error has been detected, the nonzero remainders of the two forward generators are used to determine the error pattern and location by loading them into the corresponding reverse generators in reverse order. The reverse generators are then generated with zero data input and checked for error conditions, as shown in Figure 11. The method of implementing the error correction scheme is illustrated in Figures 12 through 18. The process of locating an error starts with checking for errors in

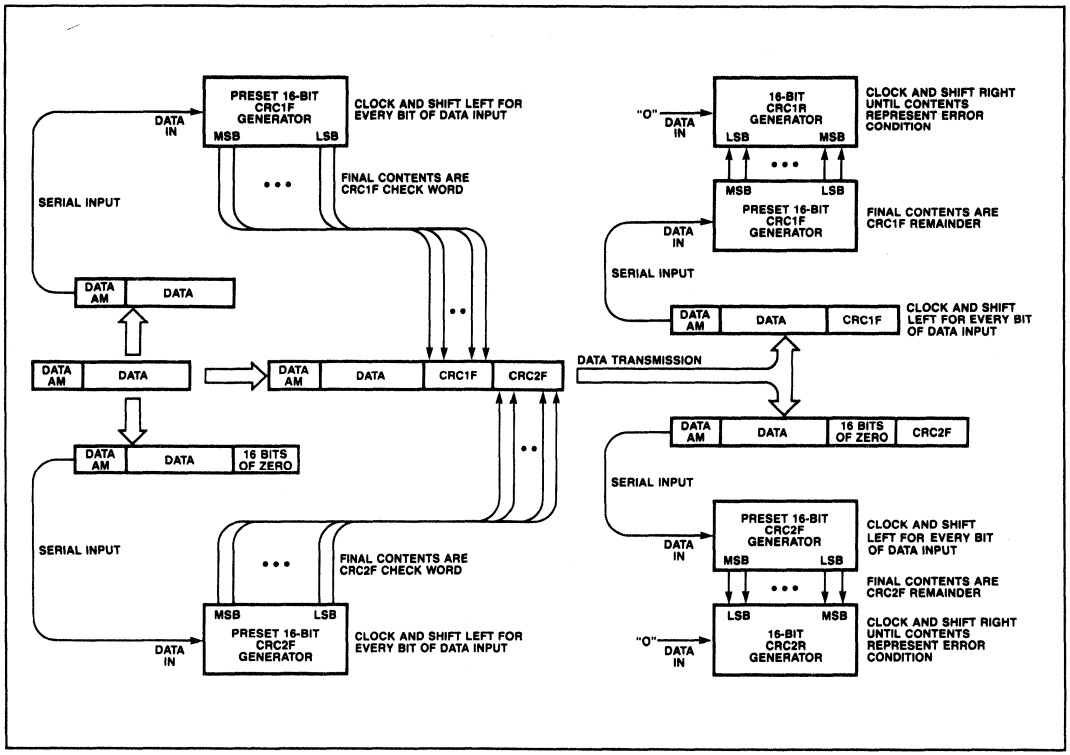


Figure 11. Data Flow

the CRC2F least significant check byte and working backwards towards the first data byte of the sector. If errors are found in the CRC check bytes, the controller will raise a CRC error status flag for the host CPU to take the appropriate action.

During the first two 8-bit parallel reverse generation cycles, only the CRC2R is generated with zero data input, the CRC1R generator remains idle. If errors occur in the CRC2F check bytes, the CRC1R should have 16 bits of zeros and the seven MSBs of CRC2R should be zeros during any of the first sixteen reverse generation clock cycles. Since the CRC2R software reverse generates one byte at a time, it is possible to miss the CRC2R error condition, because the error condition may occur in between the byte boundary. In order to solve this problem, a CHECK subroutine is written to monitor the error condition within the boundary limits. If the error condition is not found, the CHECK subroutine will swap the most significant and least significant bytes of the generator to maintain the contents of the generator before getting into the subroutine.

After the second 8-bit parallel reverse generation cycle, both the CRC1R and CRC2R are reverse generated one byte at a time until the error is found or the error is determined to be uncorrectable. An error is uncorrectable if it cannot be found within 260 8-bit reverse generation cycles. In the third 8-bit reverse generation cycle, the procedure checks for errors in the CRC1F least significant byte or errors that spanned the CRC1F byte and the CRC2F most significant byte. The error conditions are eight bits of zeros for the CRC1R most significant byte and the CRC2R least significant byte at the byte boundary. If these conditions occur, a 16-bit word is formed by concatenating the most significant byte of CRC2R with the least significant byte of CRC1R, and the CHECK subroutine, as described previously, is used to check for seven leading zeros within eight one-bit rotations towards the LSB. This condition is required to assure that errors occur in CRC1F least significant check byte or between CRC1F and CRC2F check bytes. In the fourth 8-bit reverse generation cycle, the procedure checks for errors in the CRC1F check bytes. The error conditions are 16 bits of zeros for CRC2R and seven

leading zeros in CRC1R found by the CHECK subroutine. Errors occurring in the last data byte or errors that span the CRC1F most significant check byte and the last data byte are checked in the fifth 8-bit reverse generation cycle. The error conditions are zeros in the CRC2R most significant byte, a match of the CRC1R and CRC2R least significant byte, and seven leading zeros in CRC1R found by the CHECK subroutine. The sixth to the two hundred sixtieth 8-bit reverse generation cycles check for errors in data byte 255 to the first data byte. The error conditions are a match in CRC1R and CRC2R, and seven leading zeros in CRC1R found by the CHECK subroutine.

The error conditions described in the previous sections determine when to stop the reverse generation cycle. The number of reverse generation cycles required to meet the error conditions are used to calculate the error location by the following equation:

Error location in terms of byte number = Record length in bytes + 4 CRC check bytes − number of reverse generation cycles + 1

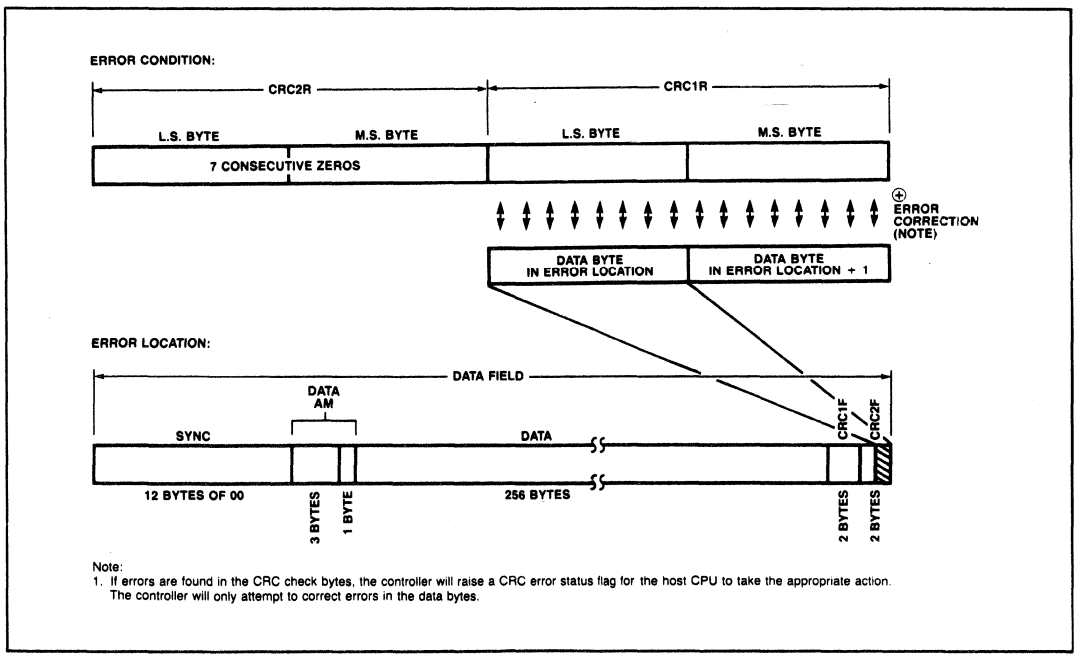
If the error location is one, then errors exist in the first data byte or between the first and second data byte of the sector. For example, if errors exist between the first and second data bytes of a 256-byte sector, then the error conditions are met in the two hundred sixtieth reverse generation cycle.

$$\text{Error location} = 256 + 4 - 260 + 1$$
$$= 1$$

The error pattern is found in the CRC1R generator. Error correction is done by XORing the data bytes in the error location and error location plus one with the CRC1R least significant and most significant byte respectively, as illustrated in Figure 11.

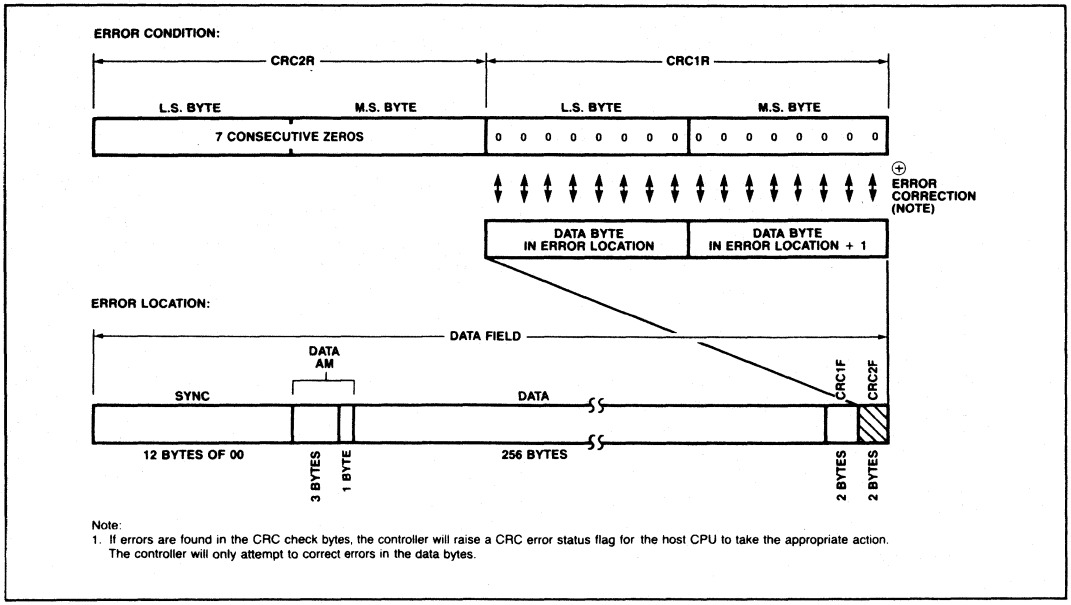
CONCLUSION

This application note describes one unique error correction algorithm for the 8X305 based floppy or hard disk controller. The algorithm can correct up to a 9-bit single burst error. The error correction procedure can be implemented with approximately 300 8X305 instructions. The worst case error correction time for a 256-byte sector is 3.0 msec.



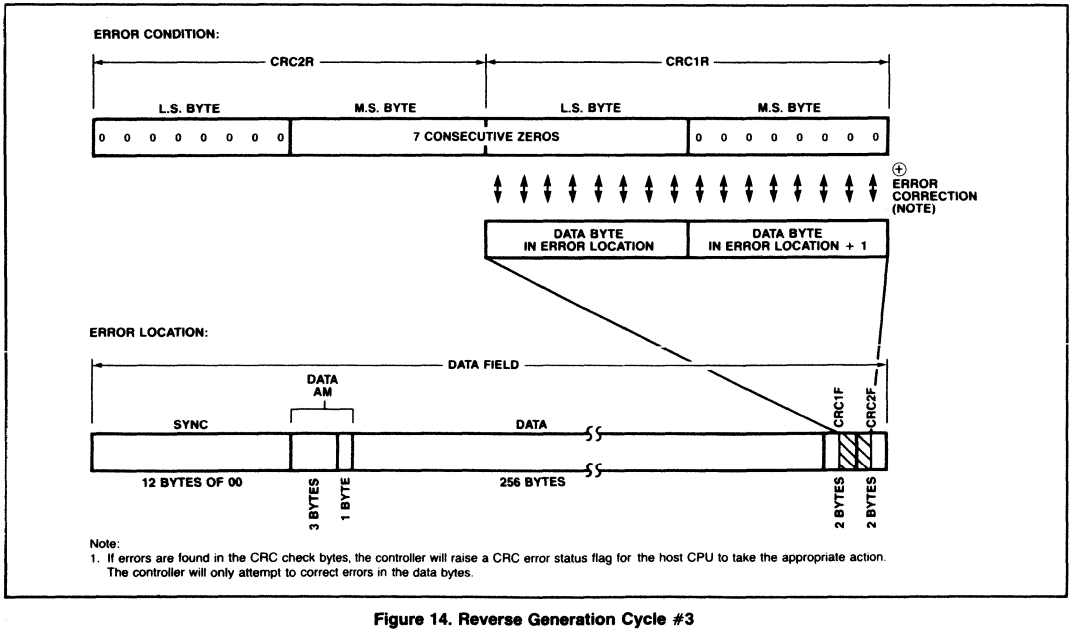
Note:
1. If errors are found in the CRC check bytes, the controller will raise a CRC error status flag for the host CPU to take the appropriate action. The controller will only attempt to correct errors in the data bytes.

Figure 12. Reverse Generation Cycle #1 (CRCIR not reverse generated)



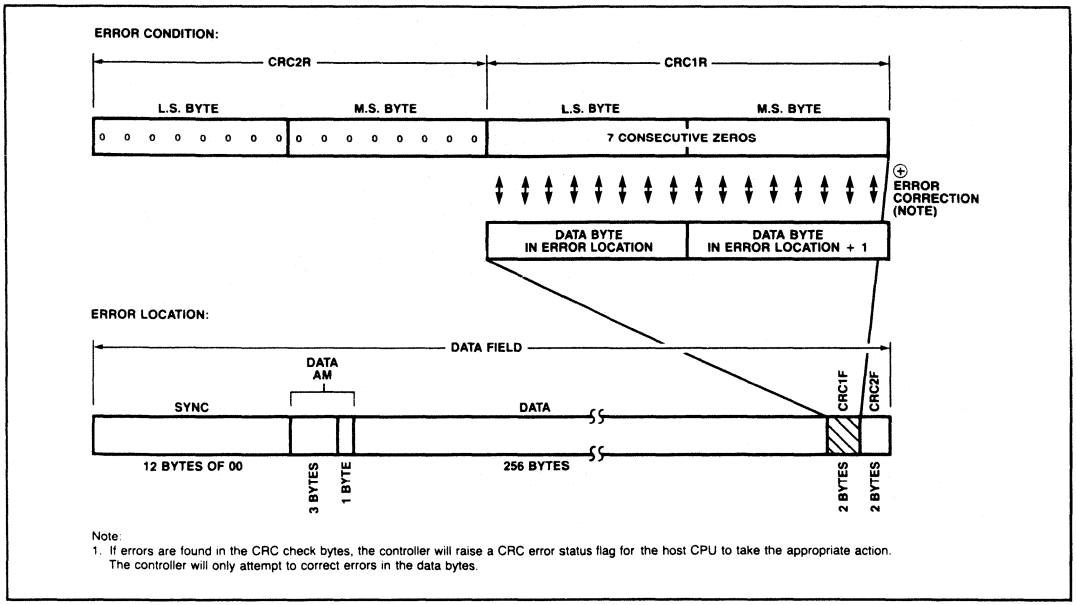
Note:
1. If errors are found in the CRC check bytes, the controller will raise a CRC error status flag for the host CPU to take the appropriate action. The controller will only attempt to correct errors in the data bytes.

Figure 13. Reverse Generation Cycle #2 (CRCIR not reverse generated)



Note:
1. If errors are found in the CRC check bytes, the controller will raise a CRC error status flag for the host CPU to take the appropriate action. The controller will only attempt to correct errors in the data bytes.

Figure 14. Reverse Generation Cycle #3



Note:
1. If errors are found in the CRC check bytes, the controller will raise a CRC error status flag for the host CPU to take the appropriate action. The controller will only attempt to correct errors in the data bytes.

Figure 15. Reverse Generation Cycle #4

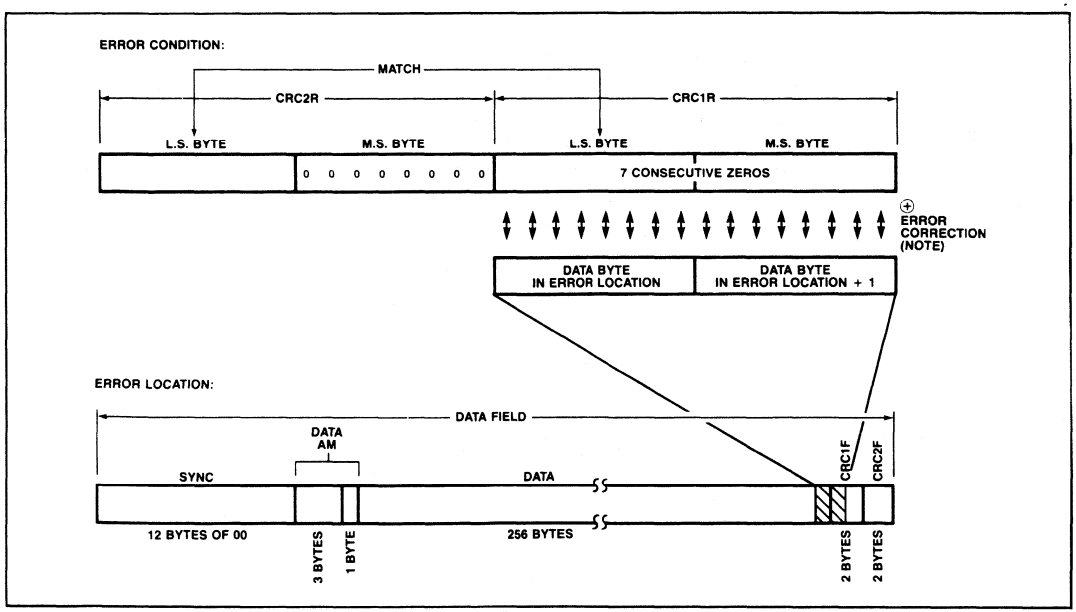


Figure 16. Reverse Generation Cycle #5

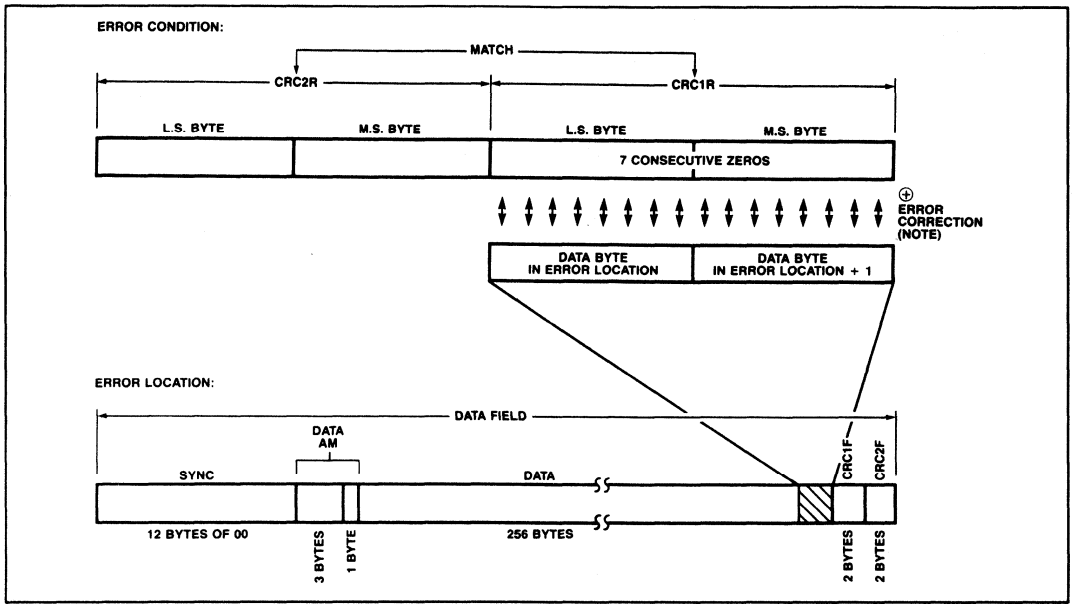


Figure 17. Reverse Generation Cycle #6

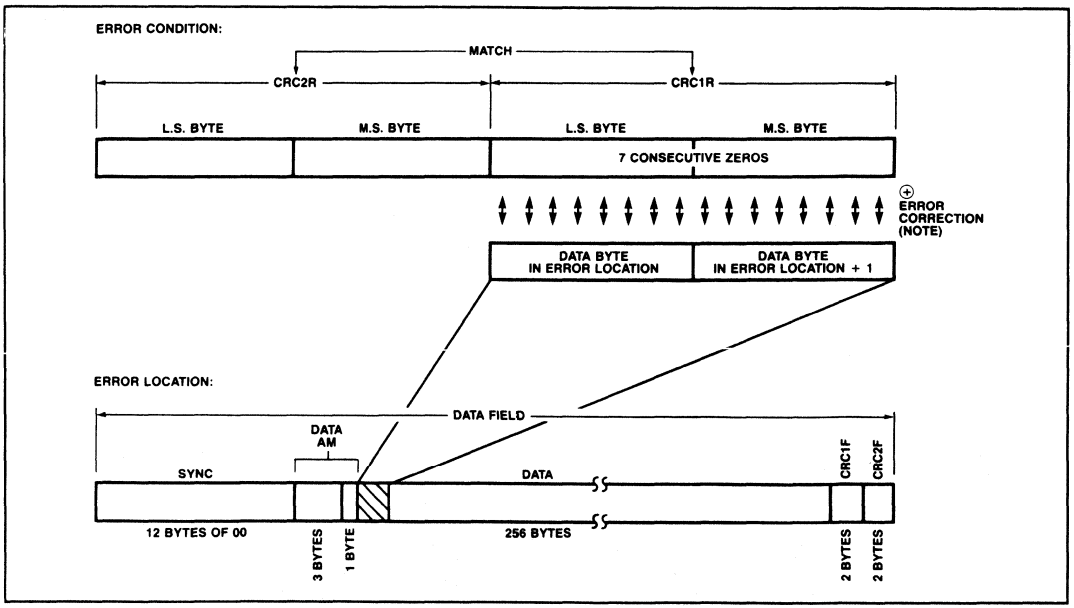
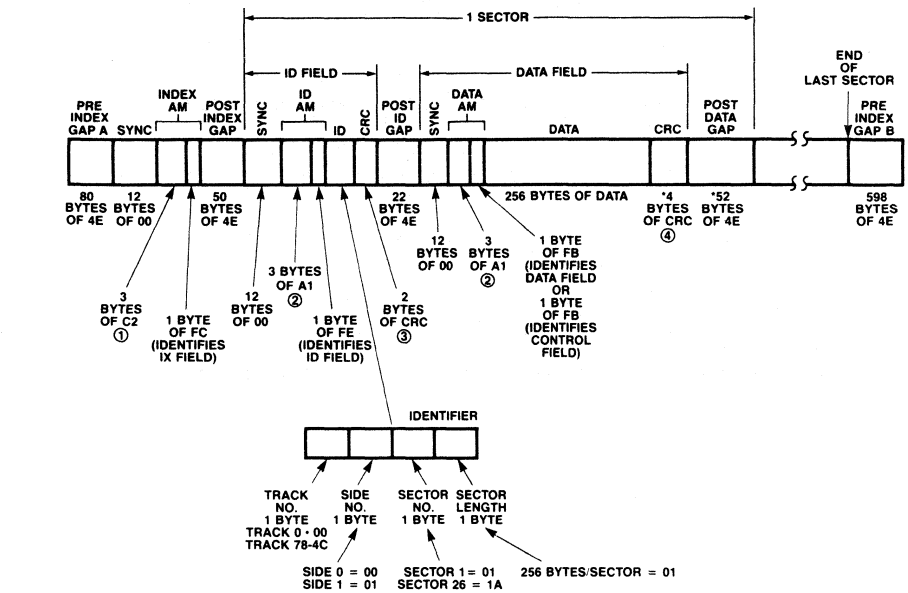


Figure 18. Reverse Generation Cycle #260



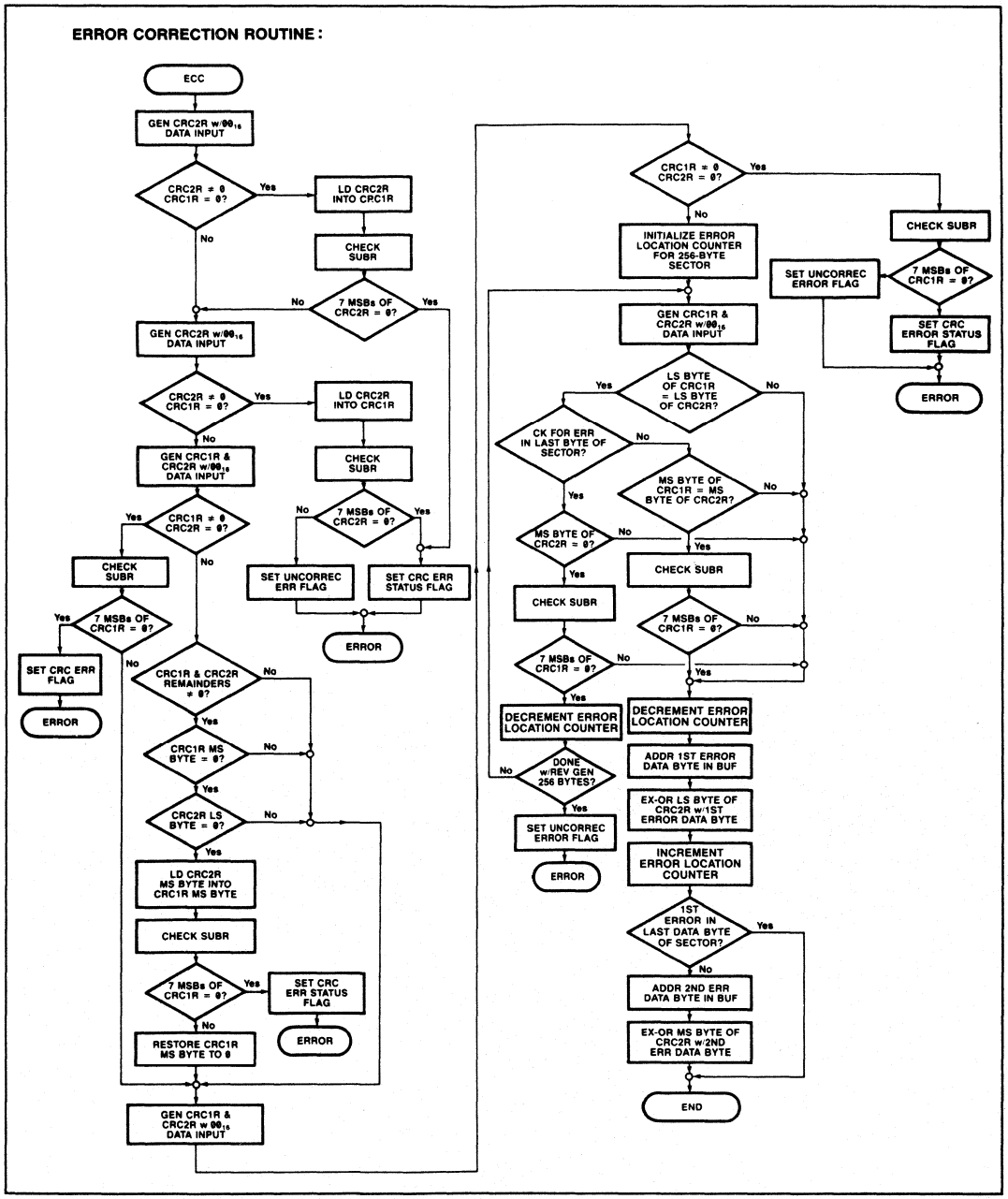
Notes:
*Indicates the modification from the IBM System 34 data track format.
1. These bytes missing clock transitions between data bits 3 and 4. Data flux transition = C2, clock flux transition = 14.
2. These bytes missing clock transitions between data bits 4 and 5. Data flux transitions = A1, clock flux transition = 0A.
3. CRC calculation starts at first byte of ID address mark and ends on the sector length byte using the CRC-CCITT polynomial.
4. The first two bytes are derived from the CRC-CCITT polynomial. The second two bytes are derived from a second 16-bit polynomial ($X^{16} + X^{13} + X^{12} + X^{10} + X^8 + X^6 + X^5 + X^4 + 1$). The CRC calculation of the 1st polynomial starts at first byte of data AM and ends on last byte of data. The CRC calculation of the 2nd polynomial includes data address mark, data bytes and two bytes of zeros.
5. There are 26 sectors of 256 bytes.

Figure 19. Modified IBM System 34 Data Track Format

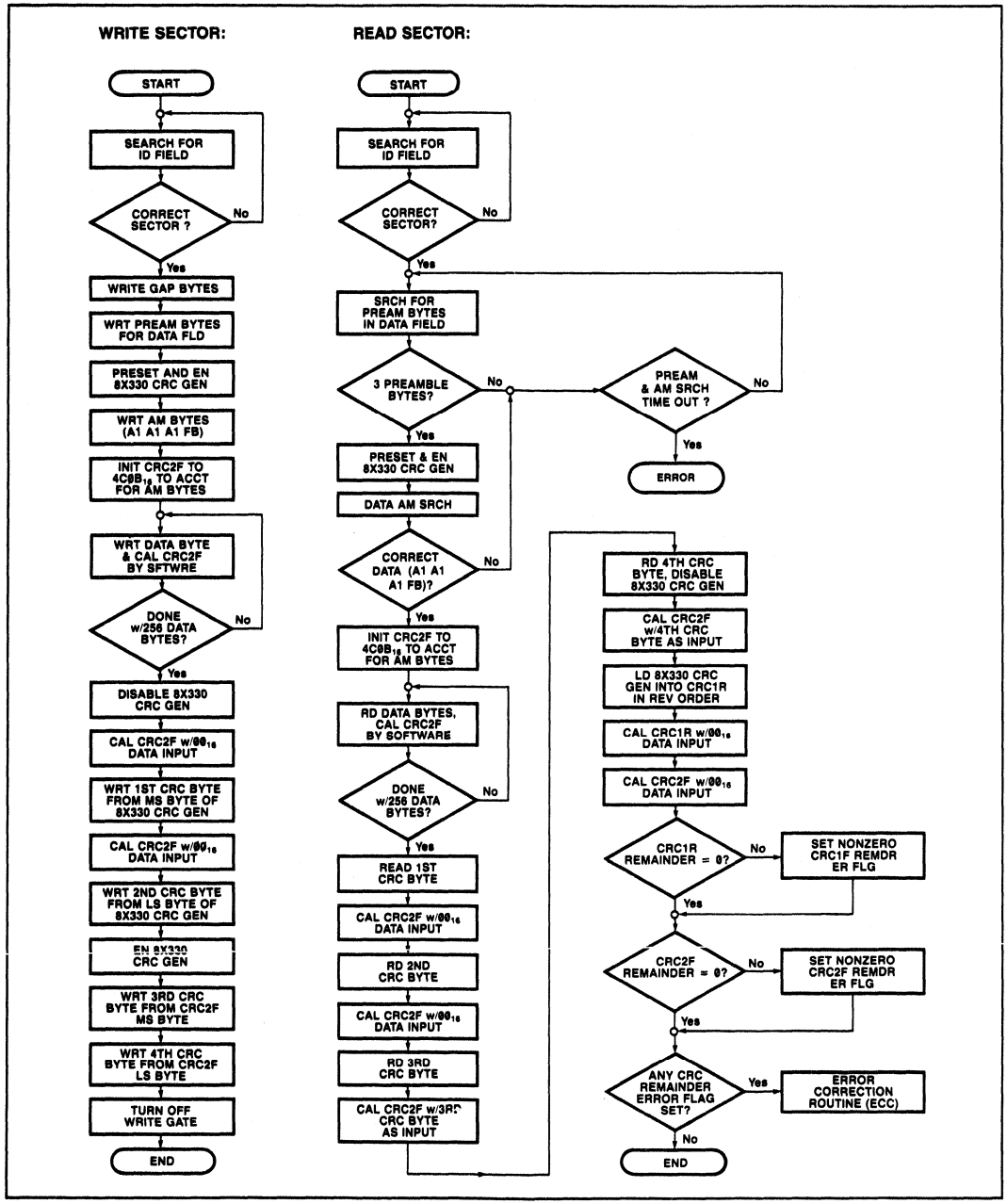
Appendix A
FLOWCHARTS

The CRC generators used to generate four CRC check bytes during the write cycle and to detect errors during the read cycle have been discussed previously and are documented by the WRITE SECTOR and READ SECTOR flowcharts that follow. One point in the READ SECTOR flowchart needs clarification. The error correction algorithm calls for the CRC1F to stop calculation after inputting the CRC1F least significant check byte during the read cycle. There is a slight modification to this procedure with the 8X330 implementation. The 8X330 internal CRC-CCITT generator (i.e., CRC1F) continues to generate up the CRC2F least significant check byte. That means two additional bytes, namely the CRC2F check bytes, are inputted to CRC1F. It is necessary to recover CRC1F remainders right after the input of the CRC1F least significant check byte as required by the error correction algorithm. This information can be recovered by loading the CRC1F remainders, which include the CRC2F check bytes in the generation, onto the CRC1R in reverse order, and then the CRC1R is generated with two bytes of zeros as shown in the READ SECTOR flowchart. At this point, the remainder of the CRC1R has the same information as if the CRC1F stopped generation after the input of the CRC1F least significant check byte.

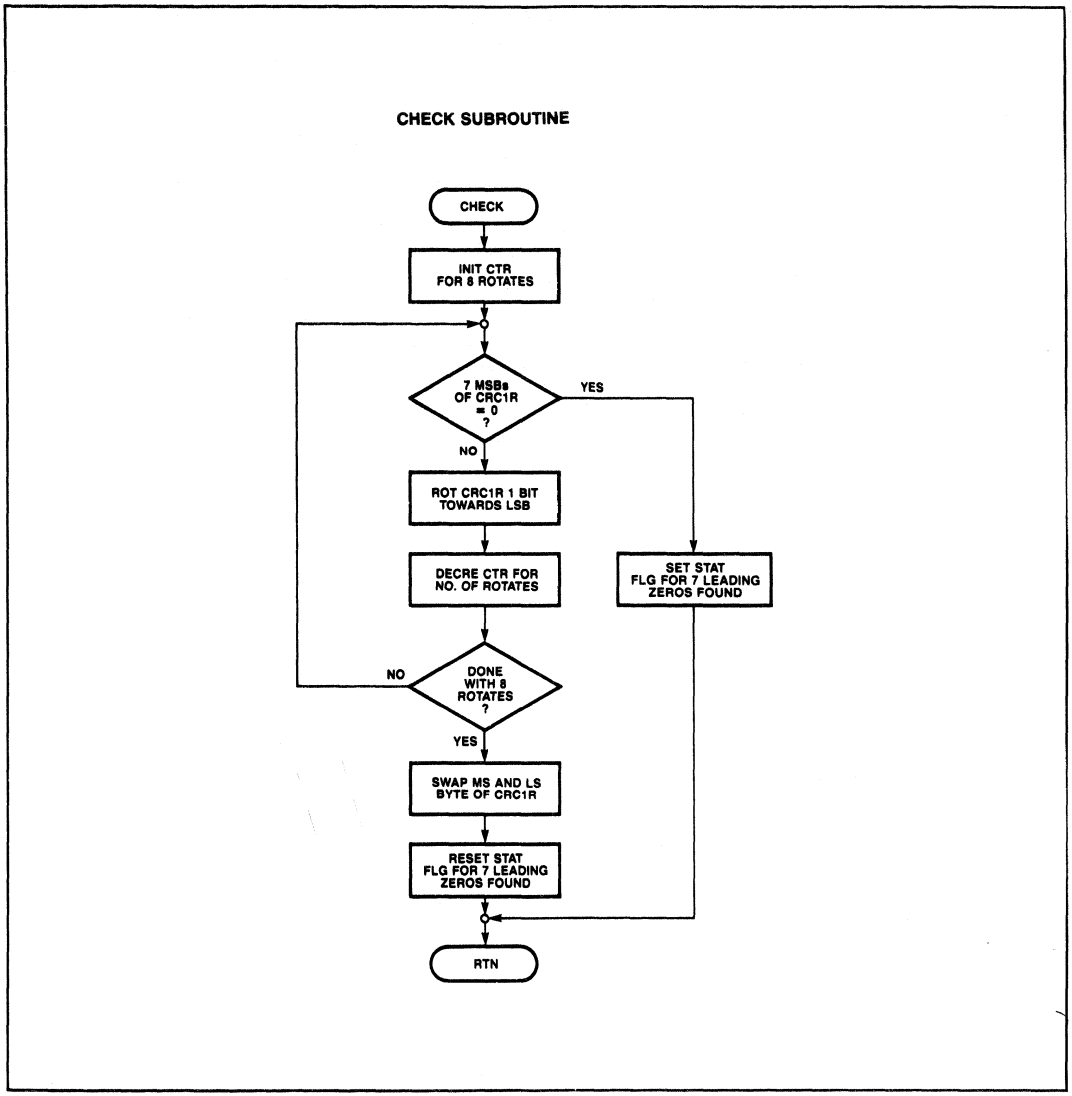
FLOWCHARTS


ERROR CORRECTION ROUTINE:
ECC
GEN CRC2R w/00₁₆ DATA INPUT
CRC2R ≠ 0 CRC1R = 0?
Yes
LD CRC2R INTO CRC1R
No
CHECK SUBR
7 MSBs OF CRC2R = 0?
No
Yes
GEN CRC2R w/00₁₆ DATA INPUT
CRC2R ≠ 0 CRC1R = 0?
Yes
LD CRC2R INTO CRC1R
No
GEN CRC1R & CRC2R w/00₁₆ DATA INPUT
CHECK SUBR
CRC1R ≠ 0 CRC2R = 0?
Yes
No
7 MSBs OF CRC2R = 0?
No
Yes
CHECK SUBR
SET UNCORREC ERR FLAG
SET CRC ERR STATUS FLAG
7 MSBs OF CRC1R = 0?
Yes
No
ERROR
SET CRC ERR FLAG
ERROR
CRC1R & CRC2R REMAINDERS ≠ 0?
No
Yes
CRC1R MS BYTE = 0?
No
Yes
CRC2R LS BYTE = 0?
No
Yes
LD CRC2R MS BYTE INTO CRC1R MS BYTE
CHECK SUBR
7 MSBs OF CRC1R = 0?
Yes
No
SET CRC ERR STATUS FLAG
ERROR
RESTORE CRC1R MS BYTE TO 0
GEN CRC1R & CRC2R w 00₁₆ DATA INPUT
CRC1R ≠ 0 CRC2R = 0?
Yes
No
CHECK SUBR
7 MSBs OF CRC1R = 0?
SET UNCORREC ERROR FLAG
ERROR
INITIALIZE ERROR LOCATION COUNTER FOR 256-BYTE SECTOR
GEN CRC1R & CRC2R w/00₁₆ DATA INPUT
LS BYTE OF CRC1R = LS BYTE OF CRC2R?
Yes
No
CK FOR ERR IN LAST BYTE OF SECTOR?
No
Yes
MS BYTE OF CRC1R = MS BYTE OF CRC2R?
No
Yes
MS BYTE OF CRC2R = 0?
No
Yes
CHECK SUBR
CHECK SUBR
7 MSBs OF CRC1R = 0?
No
Yes
7 MSBs OF CRC1R = 0?
No
Yes
DECREMENT ERROR LOCATION COUNTER
DECREMENT ERROR LOCATION COUNTER
ADDR 1ST ERROR DATA BYTE IN BUF
DONE w/REV GEN 256 BYTES?
No
Yes
EX-OR LS BYTE OF CRC2R w/1ST ERROR DATA BYTE
SET UNCORREC ERROR FLAG
ERROR
INCREMENT ERROR LOCATION COUNTER
1ST ERROR IN LAST DATA BYTE OF SECTOR?
Yes
No
ADDR 2ND ERR DATA BYTE IN BUF
EX-OR MS BYTE OF CRC2R w/2ND ERR DATA BYTE
END

FLOWCHARTS (Continued)



FLOWCHARTS (Continued)



Appendix B
PROGRAM LISTING

The error correction scheme described in this application note has been implemented in software as shown in the following program listing. This error correction program works with the existing hardware and software of the demonstration floppy disk controller as described in the Signetics 8X330 Floppy Disk Controller Manual. The first part of the error correction program is a software patch to read and write the two CRC2F check bytes for the demonstration controller program. The main error correction routine is located in the second half of the program. The error correction program is designed to work with the IBM system 34 data track format (256 data bytes per sector)—see illustration that follows the program listing. The only modification to the IBM system 34 data tract format is to add two more CRC check bytes to the end of the data field. With slight modifications, this program can also be used for other data track formats. For testing purposes additional coding has been added to the program to print out the contents of both CRC1R and CRC2R after each 8-bit reverse generation cycle. This section of codes can be deleted for the final program.

```
FILE: ECC330   OUTPUT   A   SIGNETICS CCRPORATICN                                        PAGE 001

1          PROG ECC330                  MICRCCONTROLLER CRCSS ASSEMBLER VER 2.0           PAGE    1

                                  PRCG ECC330
                            ***********************************************************************
                            *                                                                     *
                            *  ERROR CORRECTION PROGRAM.                      AUGUST, 1981.       *
                            *                                                                     *
                            *  THIS PROGRAM WORKS WITH THE THE HARDWARE AND SOFTWARE OF THE       *
                            *  DEMONSTRATION CONTROLLER AS DESCRIBED IN THE SIGNETICS 8X330 FLOPPY *
                            *  DISK CONTROLLER MANUAL.  FOUR CRC CHECK BYTES, DERIVED FROM TWO    *
                            *  SEPARATE 16-BIT POLYNOMIALS, ARE USED FOR THE DATA FIELD.          *
                            *  CRC1F= X(16)+X(12)+X(5)+1                                          *
                            *  CRC2F= X(16)+X(13)+X(12)+X(10)+X(8)+X(6)+X(5)+X(4)+1               *
                            *                                                                     *
                            ***********************************************************************
                            *
                                  LIST M,S,O
                            *     OBJ  R,B
              124 7 0       CRCM    RIV 124H,7,8                    *M.S. BYTE OF CRC2F
              124 7 0       MCRC2F RIV 124H,7,8                     *M.S. BYTE OF CRC2F
              115 7 0       CRCL    RIV 115H,7,8                    *L.S. BYTE OF CRC2F
              115 7 0       LCRC2F RIV 115H,7,8                     *L.S. BYTE OF CRC2F
              120 7 0       MCRC1R RIV 120H,7,8                     *M.S. BYTE OF CRC1R
              120 0 1       MOCRC1R RIV 120H,0,1                    *BIT 0 OF CRC1R M.S. BYTE
              120 6 1       M6CPC1R RIV 120H,6,1                    *BIT 6 OF CRC1R M.S. BYTE
              120 7 1       M7CRC1R RIV 120H,7,1                    *BIT 7 OF CRC1R M.S. BYTE
              121 7 0       LCRC1R RIV 121H,7,8                     *L.S. BYTE OF CRC1R
              121 0 1       LOCRC1R RIV 121H,0,1                    *BIT 0 OF CRC1R L.S. BYTE
              121 6 1       L6CRC1R RIV 121H,6,1                    *BIT 6 OF CRC1R L.S. BYTE
              121 7 1       L7CRC1R RIV 121H,7,1                    *BIT 7 OF CRC1R L.S. BYTE
              115 7 0       MCRC2R RIV 115H,7,8                     *M.S. BYTE OF CRC2R
              115 0 1       MOCRC2R RIV 115H,0,1                    *BIT 0 OF CRC2R M.S. BYTE
              115 6 1       M6CRC2R RIV 115H,6,1                    *BIT 6 OF CRC2R M.S. BYTE
              115 7 1       M7CRC2R RIV 115H,7,1                    *BIT 7 OF CRC2R M.S. BYTE
              124 7 0       LCRC2R RIV 124H,7,8                     *L.S. BYTE OF CRC2R
              124 0 1       LOCRC2R RIV 124H,0,1                    *BIT 0 OF CRC2R L.S. BYTE
              124 6 1       L6CRC2R RIV 124H,6,1                    *BIT 6 OF CRC2R L.S. BYTE
              124 7 1       L7CRC2R RIV 124H,7,1                    *BIT 7 OF CRC2R L.S. BYTE
              132 7 0       ST330    RIV 132H,7,8                   *8X330 STATUS REGISTER READ
              132 6 1       BYTRA    RIV 132H,6,1                   *BYTE TRANSFER ACTIVE = 0
              132 1 1       CRCE     RIV 132H,1,1                   *COMPUTE CRC=1
              114 6 2       MODU     RIV 114H,6,2                   *10B= SINGLE DENSITY, 00B= DOUBLE DEN
              137 7 0       FDAT     RIV 137H,7,8                   *DISK DATA REGISTER
              001 6 1       TDAT     RIV 1H,6,1
              001 5 1       ECRT     RIV 1H,5,1
              135 1 1       RDAT     RIV 135H,1,1
              122 7 0       G1       RIV 122H,7,8
              123 7 0       G2       RIV 123H,7,8
              124 7 0       G3       RIV 124H,7,8
              125 7 0       G4       RIV 125H,7,8
              126 7 0       G5       RIV 126H,7,8
              127 7 0       G6       RIV 127H,7,8
              136 7 0       SLEN     RIV 136H,7,8                   *POINTER FOR 2ND LEVEL SUBROUTINE
              112 0 1       PHS10    RIV 112H,0,1                   *CRC ERROR BIT IN PHY. REG. 1
               000005       DATA    EQU 5                           *DATA INPUT FOR CRC2F


FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                        PAGE 002

               000006       TEMP    EQU 6                           *TEMP REG.
1          PROG ECC330                  MICROCONTROLLER CRCSS ASSEMBLER VER 2.0           PAGE    2

               000027       LB      EQU 27H                         *LEFT BANK
               000037       RB      EQU 37H                         *RIGHT BANK
               000030       RB0     EQU 30H
               000031       RB1     EQU 31H
               000032       RB2     EQU 32H
               000033       RB3     EQU 33H
               000034       RB4     EQU 34H
               000035       RB5     EQU 35H
               000036       RB6     EQU 36H
               000037       RB7     EQU 37H
               000012       LF      EQU 012H                        *LINE FEED
               000015       CR      EQU 015H                        *CARRIAGE RETURN
                            *
               177777       POINT   SET -1
                            *
                            ***************************************************
                            *                                                 *
                            * PRINT BINARY MACRO.                             *
                            * PRINT THE CONTENT OF THE SELECT PORT IN THE RIGHT *
                            * BANK IN BINARY.                                 *
                            *                                                 *
                            ***************************************************
                            PRTBIN MACRO PRT1
                                    XMIT 1,AUX
                                    XMIT 377H-7,R6
                                    ORG 3,32
                            PRT1    MOVE R5,IVR
                                    XMIT '1',R4
                                    NZT RB,1,*+2
                                    XMIT '0',R4
                                    MOVE RB6,8,RB7
                                    CALL TTYWR
                                    ADD R6,R6
                                    NZT R6,PRT1
                                    ENDM
                            *
                            *
                            ****************************************************
                            *                                                  *
                            *  MACRO FOR SECOND LEVEL SUBROUTINE.              *
                            *  RETURN TO CALL2 BY JMP RTN2                     *
                            *                                                  *
                            ****************************************************
                            CALL2   MACRO SUBR2,LAB
                            *
                            LAB     SEL SLEN
                            POINT   SET POINT+1
                                    XMIT POINT,SLEN         *STORE RETURN JUMP INDEX IN SLEN
                                    JMP SUBR2               *GO TO 2ND LEVEL SUBROUTINE
                                    ORG RTN2+POINT+3        *INSERT THE NEXT INSTR. IN RETURN TABLE
                                    JMP LAB+3               *2ND LEVEL SUBROUTINE RETURN JUMP
                                    ORG LAB+3
```

```
FILE: ECC330    OUTPUT   A   SIGNETICS CORPORATION                                                    PAGE 003

                                    ENDM
                                ****************************************************************
1        PROG ECC330              MICROCONTROLLER CROSS ASSEMBLER VER 2.0                      PAGE    3

                                *                                                              *
                                *  PATCH JMP INSTRUCTIONS IN THE 2K PROGRAM OF THE DEMONSTRATION *
                                *  CONTROLLER TO THE READ AND WRITE CRC2F CHECK BYTES PROGRAM.  *
                                *                                                              *
                                ****************************************************************
                                    ORG 1716H
       01716   7 07000              JMP AA1                  *INITIALIZE CRC2F FOR A1,A1,A1,FB
                                    ORG 1746H
       01746   7 07010              JMP AA2                  *INITIALIZE CRC2F FOR FB
                                    ORG 1772H
       01772   7 07020              JMP AA3                  *CALCULATE CRC2F ON LINE FOR DATA FIELD
                                    ORG 1777H
       01777   7 07027              JMP AA4                  *CALCULATE CRC2F FOR 8 BITS OF '0' INPUTS
                                    ORG 2001H
       02001   6 17132              SEL ST330
                                    ORG 2003H
       02003   7 07034              JMP AA5                  *CALAULATE CRC2F FOR 8 BITS OF '0' INPUTS
                                    ORG 2010H
       02010   7 07061        P1    JMP PP2                  *READ M.S. BYTE OF CRC1 REMAINDER
                                    ORG 2017H
       02017   7 07064              JMP PP3                  *READ 3 MORE BYTES OF CRC
                                    ORG 2255H
       02255   7 07116              JMP PP4                  *INITIALIZE CRC2F FOR A1,A1,A1,FB OR FB
                                    ORG 2262H
       02262   7 07132              JMP PP5
                                    ORG 2270H
       02270   7 07140              JMP PP8
                                *********************************************************
                                *                                                       *
                                *  PATCH PROGRAM TO READ CRC2F CHECK BYTES.             *
                                *                                                       *
                                *********************************************************
                                    ORG 7000H
       07000   6 05114        AA1   XMIT 01001100B,R5        *INITIAL VALUE FOR CRC2F M.S. BYTE IN D.D
       07001   6 06013              XMIT 00001011B,R6        *INITIAL VALUE FOR CRC2F L.S. BYTE IN D.D
       07002   6 17124              SEL MCRC2F
       07003   0 05037              MOVE R5,MCRC2F           *INITIALIZE M.S. BYTE OF CRC2F IN D.D.
       07004   6 17115              SEL LCRC2F
       07005   0 06037              MOVE R6,LCRC2F           *INITIALIZE L.S. BYTE OF CRC2F IN D.D.
       07006   6 17137              SEL FDAT
       07007   7 01717              JMP 1717H
       07010   6 05177        AA2   XMIT 01111111B,R5        *INITIAL VALUE FOR CRC2F M.S. BYTE IN S.D
       07011   6 06033              XMIT 00011011B,R6        *INITIAL VALUE FOR CRC2F L.S. BYTE IN S.D
       07012   6 17124              SEL MCRC2F
       07013   0 05037              MOVE R5,MCRC2F           *INITIALIZE M.S. BYTE OF CRC2F IN S.D.
       07014   6 17115              SEL LCRC2F
       07015   0 06037              MOVE R6,LCRC2F           *INITIALIZE L.S. BYTE OF CRC2F IN S.D.
       07016   6 17137              SEL FDAT
       07017   7 01747              JMP 1747H
       07020   6 00001        AA3   XMIT 1,AUX
       07021   1 01001              ADD R1,R1
```

```
FILE: ECC330    OUTPUT   A   SIGNETICS CORPORATION                                                    PAGE 004

       07022   1 02002              ADD R2,R2
       07023   0 27005              MOVE LB,R5
       07024   6 11000              CALL CRC2F               *CALCULATE CRC2F FOR DATA FIELD
1        PROG ECC330              MICROCONTROLLER CROSS ASSEMBLER VER 2.0                      PAGE    4

       07025   7 06342
       07026   7 01774              JMP 1774H
       07027   6 17137        AA4   SEL FDAT
       07030   6 05000              XMIT 0,R5
       07031   6 11001              CALL CRC2F               *CALCULATE CRC2F WITH 8 BITS OF 0 INPUTS
       07032   7 06342
       07033   7 02000              JMP 2000H                *DURING THE FIRST CRC CHECK BYTE TIME
       07034   6 17137        AA5   SEL FDAT
       07035   6 05000              XMIT 0,R5
       07036   6 11002              CALL CRC2F               *CALCULATE CRC2F WITH 8 BITS OF 0 INPUTS
       07037   7 06342
       07040   6 17132              SEL ST330                *DURING THE SECOND CRC CHECK BYTE TIME
       07041   5 36100              NZT BYTRA,*-1            *WAIT FOR 3RD CRC CHECK BYTE
       07042   6 17137              SEL FDAT
       07043   0 37005              MOVE FDAT,R5             *CRC2F INPUT = 3RD CRC CHECK BYTE
       07044   0 05004              MOVE R5,R4               *STORE 3RD CRC BYTE IN R4
       07045   6 11003              CALL CRC2F               *CALCULATE CRC2F
       07046   7 06342
       07047   6 17132              SEL CRCE
       07050   6 31100              XMIT 0,CRCE              *DISABLE CRC1F CALCULATION
       07051   6 17132              SEL ST330
       07052   5 36111              NZT BYTRA,*-1            *WAIT FOR 4TH CRC CHECK BYTE
       07053   6 17137              SEL FDAT
       07054   0 37005              MOVE FDAT,R5             *CRC2F INPUT = 4TH CRC CHECK BYTE
       07055   0 05003              MOVE R5,R3               *STORE 4TH CRC BYTE IN R3
       07056   6 11004              CALL CRC2F               *CALCULATE CRC2F
       07057   7 06342
       07060   7 02004              JMP 2004H
       07061   6 17121        PP2   SEL LCRC1R
       07062   0 00037              MOVE AUX,LCRC1R          *LOAD M.S. BYTE OF CRC1F INTO L.S. BYTE OF
       07063   7 02013              JMP 2013H                *CRC1R
       07064   6 17120        PP3   SEL MCRC1R
       07065   0 00037              MOVE AUX,MCRC1R          *LOAD L.S. BYTE OF CRC1F INTO M.S. BYTE OF
       07066   6 11005              CALL CRC1R               *CALCULATE CRC1R WITH 8 BITS OF 0 INPUT
       07067   7 06370
       07070   6 11006              CALL CRC1R               *BECAUSE 3RD AND 4TH CRC BYTES GO THRU CRC
       07071   7 06370
       07072   0 03000              MOVE R3,AUX              *
       07073   3 37037              XOR MCRC1R,MCRC1R        *
       07074   6 17121              SEL LCRC1R
       07075   0 04000              MOVE R4,AUX              *
       07076   3 37037              XOR LCRC1R,LCRC1R        *
       07077   6 01001              XMIT 1,R1                *R1=1 IF CRC1F REMAINDERS NOT EQ. ZERO
                                    ORG 5,32
       07100   5 37004              NZT LCRC1R,*+4           *M.S. BYTE OF MCRC1F REMAINDER=0? YES, *+4
       07101   6 17120              SEL MCRC1R
       07102   5 37004              NZT MCRC1R,*+2           *L.S. BYTE OF MCRC1F REMAINDER=0? YES, *+2
       07103   6 01000              XMIT 0,R1                *R1=0 IF CRC1F REMAINDERS EQ. ZERO
       07104   6 00002              XMIT 10B,AUX
       07105   6 17124              SEL MCRC2F
```

```
FILE: ECC330   CUTPUT   A   SIGNFTICS CORPORATICN                                    PAGE 005

     206                              CRG 5,32
     207  07106  5 37012              NZT MCRC2F,LRA12       *M.S. BYTE OF MCRC2F REMAINDER=0? YES,GO
     208  07107  6 17115              SEL LCRC2F
     209  07110  5 37012              NZT LCRC2F,LRA12       *L.S. BYTE UF LCRC2F REMAINDER=0? YES,GO
1         PROG ECC330              MICROCONTROLLER CROSS ASSEMBLER VER 2.0           PAGE    5

     210  07111  7 07113              JMP *+2                *NO, GO TO *+2
     211  07112  1 01001     LRA12    ADD R1,R1              *R1=11B, BOTH CRC NCNZERO. R1=10B IF CRC2
     212  07113  5 01115              NZT R1,*+2             *CRC ERROR, GO TC ECC
     213  07114  7 02020              JMP 2020H
     214  07115  7 06C00              JMP ECC
     215                     *
     216                     *
     217                     ******************************************************
     218                     *                                                    *
     219                     *   PATCH PROGRAM TO WRITE CRC2F CHECK BYTES.        *
     220                     *                                                    *
     221                     ******************************************************
     222                     *
     223  07116  6 02000     PP4      XMIT 0,R2
     224  07117  6 17114              SEL MCDU
     225  07120  6 05177              XMIT 01111111B,R5      *INITIAL VALUE OF CRC2F M.S. BYTE IN S.D.
     226  07121  6 06033              XMIT 00011011B,R6      *INITIAL VALUE OF CRC2F L.S. BYTE IN S.D.
     227  07122  5 36225              NZT MCDU,*+3
     228  07123  6 05114              XMIT 01001100B,R5      *INITIAL VALUE CF CRC2F M.S. BYTE IN D.D.
     229  07124  6 06013              XMIT 00001011B,R6      *INITIAL VALUE OF CRC2F L.S. BYTE IN D.D.
     230  07125  6 17124              SEL MCRC2F
     231  07126  0 05037              MOVE R5,MCRC2F         *PRE-INITIALIZE M.S. BYTE CF CRC2F
     232  07127  6 17115              SEL LCRC2F
     233  07130  0 06037              MOVE R6,LCRC2F         *PRE-INITIALIZE L.S. BYTE OF CRC2F
     234  07131  7 02256              JMP 2256H
     235  07132  0 27037     PP5      MOVE LB,RB
     236  07133  0 27005              MOVE LB,R5             *R5= DATA INPUT
     237  07134  6 11007              CALL CRC2F             *CALCULATE CRC2F FOR CATA FIELD
          07135  7 06342
     238  07136  6 00001              XMIT 1,AUX
     239  07137  7 02263              JMP 2263H
     240  07140  6 01376     PP8      XMIT 376H,R1
     241  07141  6 17132              SEL ST330
     242  07142  5 36101              NZT BYTRA,*-1          *WRITE LAST DATA BYTE AND FIRST CRC BYTES
     243  07143  6 31100              XMIT 0,CRCE            *DISABLE CRC
     244  07144  6 17137              SEL FDAT
     245  07145  6 05000              XMIT 0,R5
     246  07146  6 11010              CALL CRC2F             *CALCULATE CRC2F WITH 8 BITS OF 0'S
          07147  7 06342
     247  07150  6 00001              XMIT 1,AUX             *FCR 1ST AND 2ND CRC BYTE TIME
     248  07151  1 01001              ADC R1,R1
     249  07152  5 01141              NZT R1,PP8+1
     250  07153  6 17132              SEL ST330
     251  07154  5 36113              NZT BYTRA,*-1          *WRIT 2ND CRC BYTF
     252  07155  6 31101              XMIT 1,CRCE            *ENABLE CRC
     253  07156  6 17124              SEL MCRC2F
     254  07157  0 37000              MOVE MCRC2F,AUX
     255  C7160  6 17137              SEL FDAT
     256  07161  0 0C037              MOVE AUX,FDAT
```

```
FILE: ECC330   CUTPUT   A   SIGNFTICS CORPORATICN                                    PAGE 006

     257  07162  6 17132     C200     SEL ST330
     258  07163  5 36122              NZT BYTRA,C200         *WRITE 3RD CRC BYTF
     259  C7164  6 17115              SEL LCRC2F
     260  07165  0 37000              MOVE LCRC2F,AUX
     261  07166  6 17137              SEL FDAT
1         PROG ECC330              MICROCCNTRCLLER CROSS ASSEMBLER VER 2.0              PAGE    6

     262  07167  0 00037              MOVE AUX,FDAT
     263  07170  6 01375              XMIT 375H,R1
     264  07171  6 05377              XMIT 377H,R5
     265  07172  6 17132     C202     SEL ST330
     266  07173  5 36132              NZT BYTRA,C202         *WRITE 4TH CRC BYTE + 2 BYTES OF FF
     267  07174  6 17137              SEL FDAT
     268  07175  C 05037              MOVE R5,FDAT
     269  07176  6 00001              XMIT 1,AUX
     270  07177  1 01001              ADD R1,R1
     271  07200  5 01172              NZT R1,C202
     272  07201  6 17132              SEL ST330
     273  07202  7 02300              JMP 2300H
     274                     *
     275                     ****************************************
     276                     *                                      *
     277                     *   2ND LEVEL SUBROUTINE RETURN TABLE  *
     278                     *                                      *
     279                     ****************************************
     280  07203  6 17136     RET2     SEL SLEN
     281          007203     RTN2     SET RET2
     282  07204  0 37000              MOVE SLEN,AUX
     283  07205  4 00206              XEC *+1(AUX)
     284                              CRG RTN2+23
     285                     *******************************************************
     286                     *                                                     *
     287                     *   ERROR CCRRECTION RCUTINE.                         *
     288                     *                                                     *
     289                     *******************************************************
     290                              CRG 6000H
     291                     ECC      CALL2 PRINT,P100       *PRINT CRC1R AND CRC2R
     291                     +*
     291  06000  6 17136     +P100     SEL SLEN
     291          000000     +PCINT   SET PCINT+1
     291  06001  6 37000     +        XMIT PCINT,SLEN        *STCRE RETURN JUMP INDEX IN SLEN
     291  06002  7 06234     +        JMP PRINT              *GO TO 2ND LEVEL SUBRCUTINE
     291                     +        ORG RTN2+POINT+3       *INSERT THE NEXT INSTR. IN RETURN TABLE
     291  07206  7 06003     +        JMP P100+3              *2ND LEVEL SUBROUTINE RETURN JUMP
     291                     +        ORG P100+3
     292  06003  6 11011              CALL CRC2R             *START REVERSE GENERATION
          06004  7 06407
     293                              CALL2 PRINT,P110       *PRINT CRC1R AND CRC2R
     293                     +*
     293  06005  6 17136     +P110     SEL SLEN
     293          000001     +PCINT   SET POINT+1
     293  06006  6 37001     +        XMIT POINT,SLEN        *STCRE RETURN JUMP INDEX IN SLEN
     293  06007  7 06234     +        JMP PRINT              *GO TO 2ND LEVEL SUBRCUTINE
     293                     +        CRG RTN2+POINT+3       *INSERT THE NEXT INSTR. IN RETURN TABLE
     293  07207  7 06010     +        JMP P110+3              *2ND LEVEL SUBRCUTINE RETURN JUMP
```

```
FILE: ECC330    OUTPUT    A    SIGNETICS CORPORATION                                    PAGE 007

 293                                +         CRG P110+3
 294   06010  6 00002                          XMIT 10B,AUX
 295   06011  3 01000                          XOR R1,AUX
 296   06012  5 00021                          NZT AUX,RG16          *CRC2 N.EQ TO 0 AND CRC1 =0? NO, JMP RG16
 297   06013  6 11012                          CALL TFR2T1           *YES, LOAD CRC2R INTO CRC1R
       06014  7 06466
1           PROG ECC330                    MICROCONTROLLER CROSS ASSEMBLER VER 2.0                PAGE    7
 298   06015  6 11013                          CALL CHECK            *CHECK FOR 7 LEADING ZEROS
       06016  7 06434
 299   06017  5 06021                          NZT R6,RG16           *7 LEADING ZEROS FOUND? NO, GO TO RG16
 300   06020  7 06222                          JMP CRCERR            *YES , ERROR IN CRC2
 301   06021  6 11014              RG16        CALL CRC2R
       06022  7 06407
 302                                          CALL2 PRINT,P200        *PRINT CRC1R AND CRC2R
 302                                +*
 302   06023  6 17136              +P200       SEL SLEN
 302            000002             +PCINT      SET PCINT+1
 302   06024  6 37002              +           XMIT PCINT,SLEN         *STORE RETURN JUMP INDEX IN SLEN
 302   06025  7 06234              +           JMP PRINT               *GO TO 2ND LEVEL SUBROUTINE
 302                               +           ORG RTN2+PCINT+3        *INSERT THE NEXT INSTR. IN RETURN TABLE
 302   07210  7 06026              +           JMP P200+3               *2ND LEVEL SUBROUTINE RETURN JUMP
 302                               +           ORG P200+3
 303   06026  6 00002                          XMIT 10B,AUX
 304   06027  3 01000                          XOR R1,AUX
 305   06030  5 00040                          NZT AUX,RG24          *CRC2 N.EQ 0 AND CRC1 = 0? NO, JMP RG24
 306   06031  6 11015                          CALL TFR2T1           *YES, LOAD CRC2 INTO CRC1
       06032  7 06466
 307   06033  6 11016                          CALL CHECK            *LOOK FOR 7 L.S. BITS EQUAL TO ZEROS
       06034  7 06434
 308   06035  5 06037                          NZT R6,**2            *7 LEADING ZEROS FOUND?
 309   06036  7 06222                          JMP CRCERR            *YES, ERROR IN CRC2
 310   06037  7 06233                          JMP UNCORR            *NO, UNCORRECTABLE ERROR
 311   06040  6 11017              RG24        CALL CRC1R
       06041  7 06370
 312   06042  6 11020                          CALL CRC2R
       06043  7 06407
 313                                          CALL2 PRINT,P300
 313                                +*
 313   06044  6 17136              +P300       SEL SLEN
 313            000003             +PCINT      SET PCINT+1
 313   06045  6 37003              +           XMIT PCINT,SLEN         *STORE RETURN JUMP INDEX IN SLEN
 313   06046  7 06234              +           JMP PRINT               *GO TO 2ND LEVEL SUBROUTINE
 313                               +           ORG RTN2+PCINT+3        *INSERT THE NEXT INSTR. IN RETURN TABLE
 313   07211  7 06047              +           JMP P300+3               *2ND LEVEL SUBROUTINE RETURN JUMP
 313                               +           ORG P300+3
 314   06047  6 00001                          XMIT 01B,AUX
 315   06050  3 01000                          XOR R1,AUX
 316   06051  5 00057                          NZT AUX,RG24P         *CRC2 EQ 0 & CRC1 N.EQ 0?  NO,GO TO RP24P
 317   06052  6 11021                          CALL CHECK            *YES, CHECK FOR 7 LEADING ZEROS
       06053  7 06434
 318   06054  5 06056                          NZT R6,**2            *7 LEADING ZEROS FOUND?
 319   06055  7 06222                          JMP CRCERR            *YES, ERROR IN CRC1
 320   06056  7 06116                          JMP RG32              *NO
 321   06057  6 00003              RG24P       XMIT 11B,AUX
```

```
FILE: ECC330    OUTPUT    A    SIGNETICS CORPORATION                                    PAGE 008

 322   06060  3 01000                          XOR R1,AUX
 323                                          CRC 14,32
 324   06061  5 00116                          NZT AUX,RG32          *BOTH CRC N. EQ TO 0? NO, GO TO RG32.
 325   06062  6 17120                          SEL MCRC1R            *YES, CHECK FOR ERROR SPAN CRC1 AND CRC2
 326   06063  7 06100                          CRC 16,32
 327   06100  5 37016                          NZT MCRC1R,RG32       *CRC1 M.S. BYTE =0? NO, ERROR IN DATA
 328   06101  6 17124                          SEL LCRC2R            *YES
1           PROG ECC330                    MICROCONTROLLER CROSS ASSEMBLER VER 2.0                PAGE    8
 329   06102  5 37016                          NZT LCRC2R,RG32       *CRC2 L.S. BYTE =0? NO, ERROR IN DATA
 330   06103  6 17115                          SEL MCRC2R            *YES
 331   06104  0 37000                          MOVE MCRC2R,AUX
 332   06105  6 17120                          SEL MCRC1R
 333   06106  0 00037                          MOVE AUX,MCRC1R       *LOAD CRC2 M.S. BYTE INTO CRC1 M.S. BYTE
 334   06107  6 11022                          CALL CHECK            *CHECK FOR 7 L.S. BITS EQ. ZEROS
       06110  7 06434
 335   06111  5 06113                          NZT R6,**2            *7 LEADING ZEROS FOUND? NO, ERROR IN DATA
 336   06112  7 06222                          JMP CRCERR            *YES, ERROR SPAN CRC1 AND CRC2.
 337   06113  6 00000                          XMIT 0,AUX
 338   06114  6 17120                          SEL MCRC1R
 339   06115  0 00037                          MOVE AUX,MCRC1R      *RESTORE MCRC1R VALUE BEFORE CHECK SUBR
 340   06116  6 11023              RG32        CALL CRC1R
       06117  7 06370
 341   06120  6 11024                          CALL CRC2R            *AT THE BOUNTARY OF DATA AND CRC
       06121  7 06407
 342                                          CALL2 PRINT,P400
 342                                +*
 342   06122  6 17136              +P400       SEL SLEN
 342            000004             +PCINT      SET PCINT+1
 342   06123  6 37004              +           XMIT PCINT,SLEN         *STORE RETURN JUMP INDEX IN SLEN
 342   06124  7 06234              +           JMP PRINT               *GO TO 2ND LEVEL SUBROUTINE
 342                               +           ORG RTN2+PCINT+3        *INSERT THE NEXT INSTR. IN RETURN TABLE
 342   07212  7 06125              +           JMP P400+3               *2ND LEVEL SUBROUTINE RETURN JUMP
 342                               +           ORG P400+3
 343   06125  6 00001                          XMIT 01B,AUX
 344   06126  3 01000                          XOR R1,AUX
 345   06127  5 00135                          NZT AUX,RGDATA        *CRC1 N. EQ TO 0 AND CRC2 = 0? NO, JMP RG
 346   06130  6 11025                          CALL CHECK            *CHECK FOR 7 LEADING ZEROS
       06131  7 06434
 347   06132  5 06134                          NZT R6,**2            *7 LEADING ZEROS FOUND?
 348   06133  7 06222                          JMP CRCERR            *YES, ERROR IN CRC1
 349   06134  7 06233                          JMP UNCORR            *NO, UNCORRECTIBLE ERROR
 350   06135  6 03000              RGDATA      XMIT 0,R3             *INITIALIZE BYTE COUNTER
 351   06136  6 11026              LOOP        CALL CRC1R            *REVERSE GENERATE CRC1R
       06137  7 06370
 352   06140  6 11027                          CALL CRC2R
       06141  7 06407
 353                                          CALL2 PRINT,P500
 353                                +*
 353   06142  6 17136              +P500       SEL SLEN
 353            000005             +POINT      SET PCINT+1
 353   06143  6 37005              +           XMIT PCINT,SLEN         *STORE RETURN JUMP INDEX IN SLEN
 353   06144  7 06234              +           JMP PRINT               *GO TO 2ND LEVEL SUBROUTINE
 353                               +           ORG RTN2+PCINT+3        *INSERT THE NEXT INSTR. IN RETURN TABLE
 353   07213  7 06145              +           JMP P500+3               *2ND LEVEL SUBROUTINE RETURN JUMP
```

```
FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                        PAGE 009

    353                                +        CRG P500+3
    354   06145  6 17124                        SEL LCRC2R
    355   06146  0 37000                        MOVF LCRC2R,AUX
    356   06147  6 17121                        SEL LCRC1R
    357   06150  3 37000                        XOR LCRC1R,AUX       *CCMPARE L.S. BYTE OF CRC1R AND CRC2R
    358   06151  5 00171                        NZT AUX,NCOMP        *COMPARE? NC,GC TC NCOMP
    359   C6152  6 17115                        SEL MCRC2R           *YES
    360   06153  5 03161                        NZT R3,NLDB          *R.G. FOR LAST DATA BYTE. NO, JMP NLDB
1       PROG ECC330                  MICRCCONTROLLER CROSS ASSEMBLER VER 2.0                PAGE    9

    361                                         CRG 14,32
    362   06154  5 37031                        NZT MCRC2R,NCOMP     *YES, M.S. BYTE OF CRC2R E.Q. 0?
    363   06155  6 11030                        CALL CHECK           *CHECK FOR 7 LEADING ZEROS
          06156  7 06434
    364   06157  5 06171                        NZT R6,NCOMP         *7 LEADING ZERCS FOUND? NO, JMP NCOMP
    365   06160  7 06175                        JMP CCRVTR           *YES, GO TO CORRECTION
    366   06161  0 37000                 NLDB   MOVE MCRC2R,AUX      *NOT R.G. FOR LAST DATA BYTE
    367   06162  6 17120                        SEL MCRC1R
    368   06163  3 37000                        XOR MCRC1R,AUX       *COMPARE M.S. BYTE OF CRC1R AND CRC2R
    369   06164  5 00171                        NZT AUX,NCOMP        *CCMPARE? NO, GO TO NCOMP
    370   06165  6 11031                        CALL CHECK           *CHECK FOR 7 LEADING ZEROS
          06166  7 06434
    371   06167  5 06171                        NZT R6,NCOMP         *7 LEADING ZEROS FCUND? NO, JMP NCOMP
    372   06170  7 06175                        JMP CCRVTR           *YES, GC TC CORRECTION
    373   06171  6 00377                 NCOMP  XMIT -1,AUX
    374   06172  1 03003                        ADD R3,R3            *DECREMENT BYTE COUNT
    375   06173  5 03136                        NZT R3,LOOP          *RG 256 BYTES? NO, GC TO LOOP
    376   06174  7 06233                        JMP UNCORR           *UNCORRECTABLE ERROR
    377   06175  6 04C15                 CCRVTR XMIT CR,R4
    378   06176  6 11032                        CALL TTYWR           *PRINT CR
          06177  7 06553
    379   06200  6 04012                        XMIT LF,R4
    380   06201  6 11033                        CALL TTYWR           *PRINT LF
          06202  7 06553
    381   06203  6 00377                        XMIT -1,AUX
    382   06204  1 03003                        ADD R3,R3
    383   06205  0 03007                        MOVE R3,IVL          *R3 EC. 1ST LOCATION IN ERROR
    384   06206  6 17124                        SEL LCRC2R
    385   06207  0 37000                        MOVE LCRC2R,AUX
    386   C6210  3 27027                        XOR LB,LB            *CORRECTION FOR FIRST BYTE
    387   06211  6 00001                        XMIT 1,AUX
    388   06212  1 03C03                        ADD R3,R3
    389   06213  5 03215                        NZT R3,*+2           *1ST ERROR IN LAST DATA BYTE? NO, GO TO *
    390   06214  7 06221                        JMP DCNE             *YES, GO TO DCNE
    391   06215  0 03007                        MOVE R3,IVL          *ADDRESS BUFFER
    392   06216  6 17115                        SEL MCRC2R
    393   06217  0 37000                        MOVE MCRC2R,AUX
    394   06220  3 27027                        XOR LB,LB            *CORRECTION IN 2ND BYTE
    395   06221  7 02C20                 DCNE   JMP 2020H
    396   06222  6 17112                 CRCERR SEL PHS10            *ERRCR IN CRC BYTES
    397   06223  6 30101                        XMIT 1,PHS10         *SET CRC ERROR FLAG
    398   06224  6 04015                        XMIT CR,R4
    399   06225  6 11034                        CALL TTYWR           *PRINT CR
          06226  7 06553
    400   06227  6 04012                        XMIT LF,R4
```

```
FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATICN                                        PAGE 010

    4C1   C6230  6 11C35                        CALL TTYWP           *PRINT LF
          06231  7 06553
    402   06232  7 02020                        JMP 2020H
    403   06233  7 02027                 UNCORR JMP 2027H            *UNCORRECTABLE ERROR
    404                                  *
    405                                  *
    406                                  ******************************************************************
    4C7                                  *                                                                *
    408                                  *   2ND LEVEL SUBROUTINE TO PRINT CONTENTS CF CRC1R AND CRC2R.  *
1       PROG ECC330                      MICROCONTRCLLER CROSS ASSEMBLER VER 2.0               PAGE   10

    409                                  *                                                                *
    410                                  ******************************************************************
    411   06234  6 04015                 PRINT  XMIT CR,R4
    412   06235  6 11036                        CALL TTYWR           *PRINT CR
          06236  7 06553
    413   06237  6 04012                        XMIT LF,R4
    414   06240  6 11037                        CALL TTYWR           *PRINT LF
          06241  7 06553
    415   06242  6 05120                        XMIT MCRC1R,R5
    416                                         PRTBIN PRT2          *PRINT M.S. BYTE OF CRC1R. M.S. BIT ON L.
    416   06243  6 0C001                 +      XMIT 1,AUX
    416   06244  6 06370                 +      XMIT 377H-7,R6
    416                                  +      CRG 3,32
    416   06245  0 05017                 +PRT2  MOVE R5,IVR
    416   06246  6 04061                 +      XMIT '1',R4
    416   06247  5 37111                 +      NZT RB,1,*+2
    416   06250  6 04060                 +      XMIT 'O',R4
    416   06251  0 36037                 +      MOVE RB6,8,RB7
    416   06252  6 11040                 +      CALL TTYWR
          06253  7 06553
    416   06254  1 06006                 +      ADD R6,R6
    416   06255  5 06245                 +      NZT R6,PRT2
    417   06256  6 04040                        XMIT ' ',R4
    418   06257  6 11041                        CALL TTYWR           *PRINT SPACE BETWEEN M.S. AND L.S. BYTE O
          06260  7 06553
    419   06261  6 05121                        XMIT LCRC1R,R5
    420                                         PRTBIN PRT3          *PRINT L.S. BYTE OF CRC1R. M.S. BIT ON L.
    420   06262  6 00001                 +      XMIT 1,AUX
    420   06263  6 06370                 +      XMIT 377H-7,R6
    420                                  +      CRG 3,32
    420   06264  0 05017                 +PRT3  MOVE R5,IVR
    420   06265  6 04061                 +      XMIT '1',R4
    420   06266  5 37130                 +      NZT RB,1,*+2
    420   06267  6 04060                 +      XMIT 'O',R4
    420   06270  0 36037                 +      MOVE RB6,8,RB7
    420   06271  6 11042                 +      CALL TTYWR
          06272  7 06553
    420   06273  1 06006                 +      ADD R6,R6
    420   06274  5 06264                 +      NZT R6,PRT3
    421   06275  6 04040                        XMIT ' ',R4          *PRINT 2 SPACES BETWEEN CRC1R AND CRC2R
    422   06276  6 11043                        CALL TTYWR
          06277  7 06553
    423   06300  6 04040                        XMIT ' ',R4
    424   06301  6 11044                        CALL TTYWR
```

```
FILE: ECC330   OUTPUT   A   SIGNETICS CCPPORATICN                                   PAGE 011

          06302  7 06553
     425  06303  6 05115              XMIT MCRC2R,R5
     426                              PRTBIN PRT4              *PRINT M.S. BYTE OF CRC2R. M.S. BIT ON L.
     426  06304  6 00001        +     XMIT 1,AUX
     426  06305  6 06370        +     XMIT 377H-7,R6
     426                        +     CRG 3,32
     426  06306  0 05017        +PRT4 MOVE R5,IVR
     426  06307  6 04061        +     XMIT '1',R4
     426  06310  5 37112        +     NZT RB,1,*+2
     426  06311  6 04060        +     XMIT '0',R4
1         PROG ECC330                MICROCONTRCLLER CROSS ASSEMBLER VER 2.0           PAGE   11

     426  06312  0 36037        +     MCVE RB6,8,RB7
     426  06313  6 11045        +     CALL TTYWR
          06314  7 06553
     426  06315  1 06006        +     ADD R6,R6
     426  06316  5 06306        +     NZT R6,PRT4
     427  06317  6 04040              XMIT ' ',R4
     428  06320  6 11046              CALL TTYWR               *PRINT SPACE BETWEEN M.S. AND L.S. BYTE O
          06321  7 06553
     429  06322  6 05124              XMIT LCRC2R,R5
     430                              PRTBIN PRT5              *PRINT L.S. BYTE OF CRC2R. M.S. BIT ON L.
     430  06323  6 00001        +     XMIT 1,AUX
     430  06324  6 06370        +     XMIT 377H-7,R6
     430                        +     CRG 3,32
     430  06325  0 05017        +PRT5 MOVE R5,IVR
     430  06326  6 04061        +     XMIT '1',R4
     430  06327  5 37131        +     NZT RB,1,*+2
     430  06330  6 04060        +     XMIT '0',R4
     430  06331  0 36037        +     MOVE RB6,8,RB7
     430  06332  6 11047        +     CALL TTYWR
          06333  7 06553
     430  06334  1 06006        +     ADD R6,R6
     430  06335  5 06325        +     NZT R6,PRT5
     431  06336  6 04015              XMIT CR,R4
     432  06337  6 11050              CALL TTYWR               *PRINT CR
          06340  7 06553
     433  06341  7 07203              JMP RTN2            *END OF THE PRINT ROUTINE
     434                        *
     435                        *
     436                        *
     437                        *****************************************************
     438                        *                                                   *
     439                        *  CRC2F= X(16)+X(13)+X(12)+X(10)+X(8)+X(6)+X(5)+X(4)+1 *
     440                        *  THE CRC GENFRATCR SHIFTS LEFT.                   *
     441                        *  BIT 0 CF CRCM IS THE X(15) TERM.                 *
     442                        *  BIT 7 OF CRCL IS THE X(0) TERM.                  *
     443                        *  DATA= ONE BYTE OF CATA INPUT.                    *
     444                        *  TEMP= TEMPCRARY STCRAGE.                         *
     445                        *                                                   *
     446                        *****************************************************
     447                        *
     448  06342                          PRCC CRC2F
     449  06342  6 17124        START SEL  CRCM
     450  06343  0 37000              MOVE CRCM,AUX

FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                   PAGE 012

     451  06344  3 05000              XOR  DATA,AUX            *OPERATION A
     452  06345  0 00037              MOVE AUX,CRCM
     453  06346  3 34500              XOR  RB4,5,AUX           *OPERATION B
     454  06347  3 33400              XOR  RB3,4,AUX           *OPERATION C
     455  06350  0 00037              MOVE AUX,CRCM
     456  06351  3 33433              XOR  RB3,4,RB3           *OPERATION D
     457  06352  3 32332              XOR  RB2,3,RB2           *OPERATION E
     458  06353  3 31231              XOR  RB1,2,RB1           *OPERATION F
     459  06354  0 37006              MOVE CRCM,TEMP
     460  06355  0 00037              MOVE AUX,CRCN
     461  06356  6 17115              SEL  CRCL
1         PROG ECC330                MICROCCNTROLLER CROSS ASSEMBLER VER 2.0           PAGE   12

     462  06357  3 37000              XOR  CRCL,AUX            *OPERATION K
     463  06360  0 06037              MOVE TEMP,CRCL           *L.S. BYTE RESULT IN CRCLS
     464  06361  6 17124              SEL  CRCM
     465  06362  3 33000              XOR  RB3,AUX             *OPERATION G
     466  06363  3 34000              XOR  RB4,AUX             *OPERATION H
     467  06364  3 35600              XOR  RB5,6,AUX           *OPERATION I
     468  06365  3 31037              XOR  RB1,RB7             *OPERATION J
     469  06366  0 00237              MOVE AUX,2,CRCM          *OPERATION J. M.S.  BYTE RESULT IN CRCM.
     470  06367  7 06611              RTN
     471                              END  CRC2F
     472                        *
     473                        *
     474                        *************************************************
     475                        *                                               *
     476                        *  CRC1R= X(16)+X(11)+X(4)+1                    *
     477                        *  REVERSE POLYNOMIAL OF THE CRC-CCITT          *
     478                        *  THIS CRC GENERATOR SHIFTS RIGHT WITH ZERO INPUT*
     479                        *  NOTE: M.S. BYTE CF CRC1F BECOME L.S. BYTE OF *
     480                        *      CRC1R.                                   *
     481                        *  BIT 7 OF MCRC1R IS THE X(15) TERM.           *
     482                        *  BIT 0 CF LCRC1R IS THE X(0) TERM.            *
     483                        *                                               *
     484                        *************************************************
     485                        *
     486                        *
     487  06370                          PRCC CRC1R
     488  06370  6 17120        START SEL  MCRC1R
     489  06371  0 37000              MOVE MCRC1R,AUX
     490  06372  3 32332              XOR  RB2,3,RB2           *OPERATICN A
     491  06373  0 37000              MOVE MCRC1R,AUX
     492  06374  3 33437              XOR  RB3,4,RB7           *OPERATICN B
     493  06375  0 37006              MOVE MCRC1R,TEMP
     494  06376  0 00037              MOVE AUX,MCRC1R
     495  06377  6 17121              SEL  LCRC1R
     496  06400  0 37000              MOVE LCRC1R,AUX
     497  06401  0 06037              MOVE TEMP,LCRC1R         *L.S. BYTE RESULT IN CRCLS
     498  06402  6 17120              SEL  MCRC1R
     499  06403  3 34500              XOR  RB4,5,AUX           *OPERATION C
     500  06404  3 33037              XOR  RB3,RB7             *OPERATION D
     501  06405  0 00437              MOVE AUX,4,MCRC1R        *OPERATION D. M.S.  BYTE RESULT IN CRCM.
     502  06406  7 06611              RTN
     503                              END  CRC1R
```

```
FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                     PAGE 013

                                    *
                                    *********************************************************
                                    *                                                       *
                                    *  CRC2R= X(16)+X(12)+X(11)+X(10)+X(8)+X(6)+X(4)+X(3)+1 *
                                    *  REVERSE POLYNOMIAL OF THE CRC2F.                     *
                                    *  THIS CRC GENERATOR SHIFTS RIGHT WITH ZERO DATA INPUT.*
                                    *  NOTE: M.S. BYTE OF CRC2F BECOME L.S. BYTE OF CRC2R.  *
                                    *        THIS IS DONE BY THE DECLARATION STATMENT.      *
                                    *  BIT 7 OF MCRC2R IS THE X(15) TERM.                   *
                                    *  BIT 0 OF LCRC2R IS THE X(0) TERM.                    *
                                    *                                                       *
1         PROG ECC330                  MICROCONTROLLER CROSS ASSEMBLER VER 2.0            PAGE   13

                                    *********************************************************
                                    *
                                    *
        06407                                PROC CRC2R
        06407    6 17115            START    SEL  MCRC2R
        06410    0 37000                     MOVE MCRC2R,AUX
        06411    3 33433                     XOR  RB3,4,RB3       *OPERATION A
        06412    3 32332                     XOR  RB2,3,RB2       *OPERATION B
        06413    3 31231                     XOR  RB1,2,RB1       *OPERATION C
        06414    0 37000                     MOVE MCRC2R,AUX
        06415    3 34500                     XOR  RB4,5,AUX       *OPERATION D
        06416    3 33400                     XOR  RB3,4,AUX       *OPERATION E
        06417    3 31200                     XOR  RB1,2,AUX       *OPERATION F
        06420    0 00006                     MOVE AUX,TEMP
        06421    0 37000                     MOVE MCRC2R,AUX
        06422    6 17124                     SEL  LCRC2R
        06423    3 37000                     XOR  LCRC2R,AUX      *OPERATION G
        06424    0 06037                     MOVE TEMP,LCRC2R     *L.S. BYTE RESULT IN CRCLS
        06425    6 17115                     SEL  MCRC2R
        06426    3 33000                     XOR  RB3,AUX         *OPERATION H
        06427    3 34000                     XOR  RB4,AUX         *OPERATION I
        06430    3 35600                     XOR  RB5,6,AUX       *OPERATION J
        06431    3 31037                     XOR  RB1,RB7         *OPERATION K
        06432    0 00237                     MOVE AUX,2,MCRC2R    *OPERATION K. M.S.  BYTE RESULT IN CRCM.
        06433    7 06611                     RTN
                                             END  CRC2R
                                    *
                                    *************************************************************
                                    *                                                           *
                                    *  CHECK SUBROUTINE                                         *
                                    *   1) CHECK 7 M.S. BITS OF CRC1R FOR ZEROS        *
                                    *   2) ROTATE CRC1R 1 BIT TOWARDS L.S. BIT                  *
                                    *   3) IF THE NUMBER OF ROTATE IS 8, THEN EXIT, OTHERWISE GO TO 1 *
                                    *  IF 7 LEADING ZEROS ARE FOUND, SET R6 TO ZERO.            *
                                    *                                                           *
                                    *************************************************************
                                    *
        06434                                PROC CHECK
        06434    6 02370            SHIFT    XMIT 377H-7,R2          *YES, LOAD COUNT OF 8 SHIFTS
        06435    6 17120            LOOP1    SEL MCRC1R


FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                     PAGE 014

        06436    7 06440                     ORG 4,32
        06440    5 37703                     NZT MCRC1R,7,*+3         *7 M.S.B. OF CRC1R EQ. 0?
        06441    6 06000                     XMIT 0,R6                *YES. SET R6 TO 0.
        06442    7 06465                     JMP ECHECK
        06443    0 30100                     MOVE M0CRC1,1,AUX        *NO, ROTATE CRC REG. 1 BIT TOWARDS L.S.B.
        06444    0 37736                     MOVE M7CRC1,7,M6CRC1     *
        06445    6 17121                     SEL LCRC1R               *
        06446    0 30106                     MOVE L0CRC1,1,TEMP       *
        06447    0 37736                     MOVE L7CRC1,7,L6CRC1     *
        06450    0 00137                     MOVE AUX,1,L7CRC1        *
        06451    6 17120                     SEL MCRC1R
        06452    0 06137                     MOVE TEMP,1,M7CRC1
        06453    6 00001                     XMIT 1,AUX
1        PROG ECC330                  MICROCONTROLLER CROSS ASSEMBLER VER 2.0            PAGE   14

        06454    1 02002                     ADD R2,R2
        06455    5 02035                     NZT R2,LOOP1             *DONE WITH 8 SHIFTS? NO, GO TO LOOP1
        06456    0 37006                     MOVE MCRC1R,TEMP         *RESTORE CONTENTS OF CRC1R BEFORE ENTRY
        06457    6 17121                     SEL LCRC1R               *OF THIS SUBROUTINE BY SWAPPING THE
        06460    0 37000                     MOVE LCRC1R,AUX          *MCRC1R AND LCRC1R BYTES.
        06461    0 06037                     MOVE TEMP,LCRC1R         *
        06462    6 17120                     SEL MCRC1R               *
        06463    0 00037                     MOVE AUX,MCRC1R
        06464    6 06377                     XMIT 377H,R6             *SET R6 TO ALL ONES IF NO 7 LEADING 0
        06465    7 06611            ECHECK   RTN
                                             END CHECK
                                    *
                                    *************************************************************
                                    *                                                           *
                                    *  TFR2T1 SUBROUTINE.                                       *
                                    *  THIS SUBROUTINE WILL LOAD THE CONTENTS OF CRC2R INT CRC1R. *
                                    *                                                           *
                                    *************************************************************
                                    *
        06466                                PROC TFR2T1
        06466    6 17115                     SEL MCRC2R
        06467    0 37000                     MOVE MCRC2R,AUX
        06470    6 17120                     SEL MCRC1R
        06471    0 00037                     MOVE AUX,MCRC1R
        06472    6 17124                     SEL LCRC2R
        06473    0 37000                     MOVE LCRC2R,AUX
        06474    6 17121                     SEL LCRC1R
        06475    0 00037                     MOVE AUX,LCRC1R
        06476    7 06611                     RTN
                                             END TFR2T1
                                    *
                                    ***********************************************
                                    *                                             *
                                    * TTYRE SUBROUTINE                            *
                                    * READ CRT INPUT AND STORE 7 BITS ASCII IN R4 *
                                    * G5= NO. OF DATA BITS                        *
                                    * G1= L.S. TIME DELAY COUNTER                 *
                                    * G2= M.S. TIME DELAY COUNTER                 *
                                    * R4= ASCII CRT INPUT                         *
```

```
FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                   PAGE 015

                                * DELAY IN G1 AND G2= 1 DATA BIT TIME.          *
                                * BIT TIME EQUATION = (((2XG1)+8)XG2+15)X559 NS  *
                                *                                               *
                                *************************************************

                                *
                                * BIT TIME=416.66US FOR 2400 BAUD.  BIT TIME=9.09MS FOR 110 BAUD
                                * BIT TIME=3.33MS FOR 300 BAUD.
                                      PROC TTYRE
06477                                 SEL ECRT
06477  6 17001                        XMIT 377H,R4            *CLEAR BUFFER REGISTER
06500  6 04377
06501  6 35100                        XMIT OH,ECRT            *ENABLE CRT
                                      ORG 2,32
06502  6 17135                T1     SEL RDAT
06503  5 31102                       NZT RDAT,T1             *STOP BIT?  YES, GO TO T2
1          PROG ECC330               MICROCONTROLLER CROSS ASSEMBLER VER 2.0                  PAGE   15

06504  6 17135                T2     SEL RDAT
06505  5 31107                       NZT RDAT,T3             *START BIT?  YES, GO TO T3
06506  7 06506                       JMP T2                  *NO, GO TO T2
06507  6 17126                T3     SEL G5
06510  6 00370                       XMIT 377H-7,AUX
06511  0 00037                       MOVE AUX,G5             *LOAD COUNT OF 8.
06512  6 00331                       XMIT 331H,AUX           **DELAY FOR ONE AND A HALF BIT TIME
06513  6 17123                T6     SEL G2                  *AUX=331H FOR 300 BAUD
06514  0 00037                       MOVE AUX,G2             *LOAD M.S. TIME DELAY COUNTER
06515  7 06534                       JMP T5
06516  6 17135                T11    SEL RDAT
06517  0 31100                       MOVE RDAT,AUX
06520  3 04104                       XCR  R4(1),R4           *STORE DATA BIT IN R4
06521  6 00001                       XMIT 1,AUX
06522  6 17126                       SEL G5
06523  1 37037                       ADD G5,G5               *INCR. COUNT OF DATA BITS
06524  6 00346                       XMIT 346H,AUX           **DELAY ONE DATA BIT TIME
06525  7 06513                       JMP T6                  *AUX=346H FOR 300 BAUD
06526  6 17001                T12    SEL ECRT
06527  6 35101                       XMIT 1H,ECRT            *DISABLE CRT
06530  6 00376                       XMIT 376H,AUX
06531  2 04004                       AND R4,R4               *MASK 7 M.S. BITS
06532  0 04104                       MOVE R4(1),R4           *ROTATE RIGHT ONE PLACE
06533  7 06611                       RTN
06534  6 17122                T5     SEL G1
06535  6 00004                       XMIT 4H,AUX             **AUX=4 FOR 300 BAUD
06536  0 00037                       MOVE AUX,G1
06537  6 00001                       XMIT 1,AUX
                                      CRG 2,32
06540  1 37037                T4     ADD G1,G1               *START TIME DELAY FOR 1 DATA BIT TIME
06541  5 37C00                       NZT G1,T4
06542  6 17123                       SEL G2
06543  1 37037                       ADD G2,G2
                                      CRG 3,32
06544  5 37006                       NZT G2,T100             *END OF TOME DELAY?  NO, GO BACK TO T5
06545  7 06547                       JMP **2                 *YES, END OF 1 DATA BIT TIME DELAY
06546  7 06534                T100   JMP T5
06547  6 17126                       SEL G5
06550  5 37C12                       NZT G5,**2              *DONE WITH ALL 8 DATA BIT?  NO,GO TO T11
```

```
FILE: ECC330   OUTPUT   A   SIGNETICS CORPORATION                                   PAGE 016

06551  7 06526                       JMP T12                 *YES, GO TO T12
06552  7 06516                       JMP T11
                                      END TTYRE
                                *
                                *
                                *************************************************
                                *                                               *
                                *                                               *
                                * TTYWR SUBROUTINE                              *
                                * OUTPUT ASCII DATA IN R4 TO CRT                *
                                * G2= M.S. TIME DELAY COUNTER                   *
                                * G1= L.S. TIME DELAY COUNTER                   *
                                * G5= COUNT OF BIT TIME                         *
                                * R4= ASCII DATA                                *
                                *                                               *
1          PROG ECC330                  MICROCONTROLLER CROSS ASSEMBLER VER 2.0                  PAGE   16

                                *                                               *
                                *************************************************
                                *
06553                                 PROC TTYWR
06553  6 17001                        SEL TDAT
06554  6 36100                        XMIT OH,TDAT            *START BIT
06555  6 17126                        SEL G5
06556  6 00377                        XMIT 377H-8,AUX
06557  0 00037                        MOVE AUX,G5             *LOAD COUNT OF 9 BIT TIME
06560  6 00346                T21     XMIT 346H,AUX           **DELAY ONE BIT TIME,
06561  6 17123                        SEL G2                  *AUX=346H FOR 300 BAUD
06562  0 00037                        MOVE AUX,G2             *LOAD M.S. TIME DELAY COUNTER
                                      ORG 9,32
06563  6 17122                T25     SEL G1                  *START OF DELAY LOOP
06564  6 00003                        XMIT 3H,AUX             **AUX=3H FOR 300 BAUD
06565  0 00037                        MOVE AUX,G1             *LOAD L.S. TIME DELAY COUNTER
06566  6 00001                        XMIT 1,AUX
06567  1 37C37                T24     ADD G1,G1
06570  5 37C27                        NZT G1,T24
06571  6 17123                        SEL G2
06572  1 37C37                        ADD G2,G2
06573  5 37023                        NZT G2,T25
06574  6 17126                        SEL G5                  *END OF DELAY LOOP
                                      ORG 3,32
06575  5 37C37                        NZT G5,C39
06576  7 06611                        RTN
06577  1 37037                C39     ADD G5,G5               *INCR. DATA BIT COUNT
                                      ORG 3,32
                                      NZT G5,C36
                                      JMP C37
06602  6 17001                C36     SEL TDAT
06603  0 04136                        MOVE R4,TDAT            *SHIFT OUT L.S.B. OF R4 TO TDAT
06604  0 04104                        MOVE R4(1),R4           *RIGHT ROTATE ONE PLACE
06605  7 06560                        JMP T21
06606  6 17001                C37     SEL TDAT
06607  6 36101                        XMIT 1,TDAT             *XMIT STOP BIT
06610  7 06560                        JMP T21
                                      END TTYWR
```

```
FILE: ECC330    OUTPUT    A    SIGNETICS CORPORATICN                                    PAGE 017

      716                             END ECC330

   RETURN TABLE
              06611   4  11212
              06612   7  07C26
              06613   7  07033
              06614   7  07C40
              06615   7  07C47
              06616   7  07060
              06617   7  07070
              06620   7  07C72
              06621   7  07136
              06622   7  07150
1             PROG ECC330              MICROCONTROLLER CRCSS ASSEMBLER VER 2.0          PAGE   17
              06623   7  06C05
              06624   7  06015
              06625   7  06017
              06626   7  06023
              06627   7  06033
              06630   7  06035
              06631   7  06042
              06632   7  06044
              06633   7  06C54
              06634   7  06111
              06635   7  06120
              06636   7  06122
              06637   7  06132
              06640   7  06140
              06641   7  06142
              06642   7  06157
              06643   7  06167
              06644   7  06200
              06645   7  06203
              06646   7  06227
              06647   7  06232
              06650   7  06237
              06651   7  06242
              06652   7  06254
              06653   7  06261
              06654   7  06273
              06655   7  06300
              C6656   7  06303
              06657   7  06315
              06660   7  06322
              06661   7  06334
              06662   7  06341

   ASSEMBLER ERRORS =      0
1             PROG ECC330              MICROCCNTRCLLER CRCSS ASSEMBLER VER 2.0          PAGE   18


FILE: ECC330    OUTPUT    A    SIGNETICS CCRPCRATICN                                    PAGE 018

                               SYMBOL TABLE

   *  1

   AA1      007000     AA2      007010     AA3      007020     AA4      C07027
   AA5      007034     AUX      000000     BYTRA    013261     C200     007162
   C202     007172     CALL2    000015     CHECK    006434     CORVTR   006175
   CR       000015     CRC1R    006370     CRC2F    006342     CRC2R    C06407
   CRCE     013211     CRCERR   006222     CRCL     011570     CRCM     012470
   DATA     000005     DONE     006221     ECC      006000     ECC330   C00000
   ECRT     000151     FCAT     013770     G1       012270     G2       012370
   G3       012470     G4       012570     G5       012670     G6       012770
   IVL      000007     IVP      000017     LOCRC1   012101     L0CRC2   012401
   L6CRC1   012161     L6CRC2   012461     L7CRC1   012171     L7CRC2   012471
   LB       000027     LCRC1R   012170     LCRC2F   011570     LCRC2R   012470
   LF       000012     LOOP     006136     LRA12    007112     M0CRC1   012001
   M0CRC2   011501     M6CRC1   012061     M6CRC2   011561     M7CRC1   012071
   M7CRC2   011571     MCRC1R   012070     MCRC2F   012470     MCRC2R   011570
   MODU     011462     NCCMP    006171     NLDB     006161     OVF      000010
   P1       002010     PHS10    011201     POINT    000005     PP2      007061
   PP3      007064     PP4      007116     PP5      007132     PP8      007140
   PRINT    006234     PRTBIN   000001     R0       000000     R1       000001
   R11      000011     R2       000002     R3       000003     R4       000004
   R5       000005     R6       000006     RB       000037     RB0      000030
   RB1      000031     RB2      000032     RB3      000033     RB4      000034
   RB5      000035     RB6      000036     RB7      000037     RDAT     C13511
   RET2     007203     RG16     006021     RG24     006040     RG24P    006057
   RG32     006116     RGDATA   006135     RTN2     007203     SLEN     013670
   ST330    013270     TCAT     000161     TEMP     000006     TFR2T1   006466
   TTYRE    006477     TTYWR    006553     UNCCRR   006233

   *  2

   P100     006000

   *  3

   P110     006005

   *  4

   P200     006023

   *  5

   P300     006044

   *  6

   P400     006122

   *  7

1
```

```
P50C    006142
*  8
PRT2    006245
*  9
PRT3    006264
* 10
PRT4    006306
* 11
PRT5    006325
* 12
START   006342
* 13
START   006370
* 14
START   006407
* 15
ECHECK  006465      LCCP1   006435      SHIFT   006434
* 16

* 17
T1      006502      T100    006546      T11     006516      T12     006526
T2      006504      T3      006507      T4      006540      T5      006534
T6      006513
* 18
C36     006602      C37     006606      C39     006577      T21     006560
T24     006567      T25     006563
1
```

Software support

MCCAP 8X300/8X305 CROSS ASSEMBLER PROGRAM

Originally published by Signetics January 1984

The MicroController Cross Assembler Program (MCCAP) has been developed to support the Signetics 8X300/8X305 MicroController. MCCAP provides many powerful features including macros, automatic subroutine handling, conditional assembly and extended instructions. These features significantly reduce the time required to compose and assemble MicroController programs. When combined with standard assembler features such as mnemonic op-codes and address labels, these extended features make MCCAP a powerful programming tool.

As input, MCCAP accepts source code written according to the rules presented in this manual. After assembling the source input, MCCAP produces an assembly listing and machine-readable object module.

MCCAP is written in ANSI standard FORTRAN IV and is available on the more popular timesharing services. MCCAP is also available as a fully supported product from Signetics for use on a user's in-house system.

MCCAP 8X300/8X305 CROSS ASSEMBLER PROGRAM

MCCAP, the crossassembler program for the 8X300 and 8X305 Micro-Controllers, is supplied as a 9-track magnetic tape containing FORTRAN IV source code for the crossassembler program. For compatibility with various computer systems, the tape is available in various combinations of density and data encoding. To order, use the following part numbers:

NUMBER	DENSITY	ENCODING
8X300 AS1-3SS	1600 BPI	ASCII
8X300 AS1-4SS	1600 BPI	EBCDIC

8X300/8X305 CROSS ASSEMBLER

INTRODUCTION

The 8X300AS2 runs on an Intel Intellec™ Microcomputer Development System with 64K memory under the contorl of ISIS II operating system.

The 8X300AS2 is composed of a two-pass crossassembler program and a PROM formatter overlay. Both programs are written entirely in Intel 8080 Assembly language, and are assembled on the Intel 8080/8085 Macro Assembler version 4.0, linked and located to execute in overlay.

It assembles both 8X300 and 8X305 programs. Also needed is at least one single or double density disk drive with the PROM formatter overlay always residing in drive zero.

The 8X300AS2 software is contained on three diskettes. Disks 1 and 2 contain the Single Density version and disk 3 contains the Double Density version. Both versions will be shipped when ordered under this part number.

Special purpose circuits

CRC GENERATOR/CHECKER

Originally published by Signetics January 1984

PRELIMINARY

DESIGN FEATURES

- **TTL inputs/outputs**
- **12MHz (Max) data rate**
- **Separate preset/reset controls**
- **SDLC specified pattern match (8X01A only)**
- **Automatic right justification**
- **Pin-for-pin compatibility and functionally identical with 8X01 (8X01A only)**
- **V_{CC} = 5V**
- **14-Pin DIP**

USE AND APPLICATION

- **Floppy and other disk systems**
- **Digital cassette and cartridge systems**
- **Data communication systems**

PRODUCT DESCRIPTION

The CRC Generator/Checker (8X01A or 9401) provides error-correction capabilities for digital systems that handle serial data. The two parts differ in that the 8X01A provides Synchronous Data Link Control (SDLC).

The serial data stream is divided by a selected polynomial; the remainder resulting from this algebraic process is transmitted at the end of the data stream as a Cyclic Redundancy Check Character (CRCC). At the receiving end, the same calculation is performed on the data. If the received message is error-free, the calculated remainder should satisfy a predetermined pattern. In most cases, the remainder is zero; however, where SDLC protocols (8X01A only) are used, the correct remainder is 1111000010111000 (X^0-X^{15}).

Eight polynomials are provided and any of these can be selected via a 3-bit control bus. Popular polynomials, such as CRC-16 and CCITT are implemented and the one selected can be programmed to start with all zeroes or all ones. Right justification for polynomials of degree less than 16 is automatic.

FUNCTIONAL OPERATION

8X01A and 9401

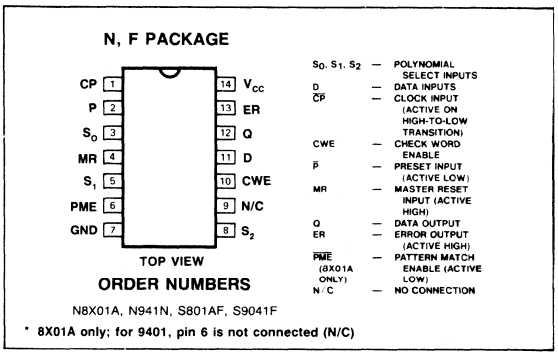
The CRC Generator/Checker circuit provides a means of detecting errors in a serial data communications environment. A binary message can be interpreted as a binary polynomial H(x). This polynomial can be divided by a generator polynomial P(x) such that H(x) = P(x) Q(x) + R(x) whereby Q(x) is the quotient and R(x) is the remainder. During transmission, the remainder is appended to the end of the message as check bits. For a given message, a unique remainder is generated. Hardware implementation of division is simply a feedback shift register with Exclusive-OR gating. Subtraction and addition in modulo 2 is implemented by the Exclusive-OR function. The number of shift register stages is equal to the degree of the divisor polynomial.

The accompanying truth table defines the polynomials implemented in the CRC circuit. Each polynomial can be selected via control inputs S_0, S_1 and S_2. To generate the check bits, the data stream is entered via the Data (D) input, using the high to low transition of the Clock ($\overline{CP}$) input. This data is gated with the most significant output (Q) of the shift register which, in turn, controls the exclusive OR gates. The Check Word Enable (CWE) must be held high while the data is being entered. After the last data bit is entered, the CWE is brought low and the check bits are shifted out of the register and appended to the data bits using external gating—see Check Word Generation diagram.

To check an incoming message for errors, both the data and check bits are entered through the "D" input with the CWE input held high. The 8X01A while not in the data path, monitors the message. After the last check bit is entered, in the 8X01A, the ERror output is made valid by a high-to-low transition of $\overline{CP}$. If no error is detected during the data transmission, all bits of the internal register are low and the ERror output is also low; if an error is detected, it is reflected by the bit pattern and the ERror output is high. The ERror output status remains valid until the next high-to-low transition of $\overline{CP}$ or until initialized by the preset ($\overline{P}$) or reset (MR) functions. The $\overline{PME}$ line must be high if the ERror output is used to indicate an all-zero result.

A high level applied to the Master Reset (MR) input asynchronously clears the shift register. A low level applied to the Preset (P) input asynchronously sets all bits to the appropriate state if the control-code inputs (S_0, S_1, and S_2) specify a 16-bit polynomial. In the

8X01A & 9401 PACKAGE/PIN DESIGNATOR

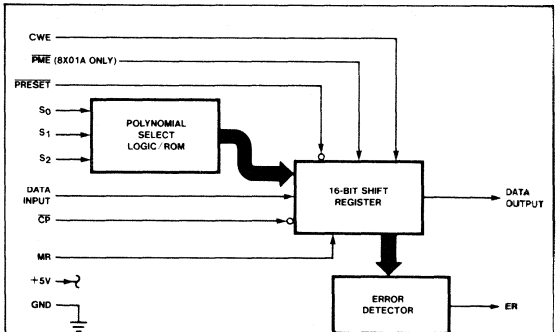


ORDER NUMBERS

N8X01A, N941N, S801AF, S9041F

* **8X01A only; for 9401, pin 6 is not connected (N/C)**

BLOCK DIAGRAM OF 8X01A & 9401



NOTE

Refer to Truth Table on next page for selection of polynomials.

PRELIMINARY

FUNCTIONAL OPERATION (cont'd)

case of check polynomials that are 8-or-12 bits in length, only the most significant 8-or-12 bits of the shift register are set; all remaining bits are cleared.

8X01A ONLY

For data communications using the Synchronous Data Link Control (SDLC) protocol, the 8X01A is preset to an all-ones configuration before any accumulation is done; this applies to both transmitting and receiving modes of operation. Using SDLC, the check sum shifted out of the 8X01A must be inverted.

During the receiving mode, a special pattern of 1111000010111000 (X^0-X^{15}) is used in place of all-zeroes to check for a valid message. The Pattern Match Enable pin allows the user to select this option. If $\overline{PME}$ is low during the last bit time of the message, the ERror output is low providing the result matches the special pattern; if an error occurs, ER is high.

TRUTH TABLE

SELECT CODE			POLYNOMIAL	REMARKS
S_2	S_1	S_0		
L	L	L	$X^{16} + X^{15} + X^2 + 1$	CRC-16
L	L	H	$X^{16} + X^{14} + X + 1$	CRC-16 REVERSE
L	H	L	$X^{16} + X^{15} + X^{13} + X^7 + X^4 + X^2 + X^1 + 1$	
L	H	H	$X^{12} + X^{11} + X^3 + X^2 + X + 1$	CRC-12
H	L	L	$X^8 + X^7 + X^5 + X^4 + X + 1$	
H	L	H	$X^8 + 1$	LRC-8
H	H	L	$X^{16} + X^{12} + X^5 + 1$	CRC-CCITT
H	H	H	$X^{16} + X^{11} + X^4 + 1$	CRC-CCITT REVERSE

RECOMMENDED OPERATING CONDITIONS

PARAMETER		LIMITS			UNIT
		Min	Typ	Max	
V_{CC}	Supply voltage	4.75	5.0	5.25	V
$\overline{CP}$	Clock input	0		12	MHz

DC ELECTRICAL CHARACTERISTICS FOR 8X01A

PARAMETER	DESCRIPTION	TEST CONDITIONS[1]	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{IH}	Input high voltage		2.0			2.0			V
V_{IL}	Input low voltage				0.8			0.7	V
V_{IC}	Input clamp diode voltage	V_{CC} = Min, I_{IN} = −18mA		−0.9	−1.5		−0.9	−1.5	V
V_{OH}	Output high voltage	V_{CC} = Min, I_{OH} = −400μA	2.7	3.4		2.4	3.4		V
V_{OL}	Output low voltage	V_{CC} = Min, I_{OL} = 4.0mA		0.35	0.4		0.35	0.4	V
		V_{CC} = Min, I_{OL} = 8.0mA		0.45	0.5		—	—	V
I_{IL}	Input low current	V_{CC} = Max, V_{IN} = 0.4V		−0.22	−0.36		−0.22	−0.36	mA
I_{IH}	Input high current	V_{CC} = Max, V_{IN} = 2.7V			20			20	μA
I_{IH}	Max input current	V_{CC} = Max, V_{IN} = 7V			0.1			0.1	mA
I_{OS}	Output short circuit current	V_{CC} = Max, V_{OUT} = 0V[2]	−10		−42	−10		−42	mA
I_{CC}	Supply current	V_{CC} = Max, inputs open		60	110		60	110	mA

DC ELECTRICAL CHARACTERISTICS FOR 9401

PARAMETER	DESCRIPTION	TEST CONDITIONS[1]	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
			Min	Typ	Max	Min	Typ	Max	
V_{IH}	Input high voltage	Guar. input high voltage	2.0			2.0			V
V_{IL}	Input low voltage	Guar. input low voltage			0.8			0.7	V
V_{IC}	Input clamp diode voltage	V_{CC} = Min I_{IN} = −18mA		−0.9	−1.5		−0.9	−1.5	V
V_{OH}	Output high voltage	V_{CC} = Min, I_{OH} = −400μA	2.4	3.4		2.4	3.4		V
V_{OL}	Output low voltage	V_{CC} = Min, I_{OL} = 4.0mA		0.35	0.4		0.35	0.4	V
		V_{CC} = Min, I_{OL} = 8.0mA		0.45	0.5		—	—	V
I_{IL}	Input low current	V_{CC} = Max, V_{IN} = 0.4V		−0.22	−0.36		−0.22	−0.36	mA
I_{IH}	Input high current	V_{CC} = Max, V_{IN} = 2.7V		1.0	40		1.0	40	μA
		V_{CC} = Max, V_{IN} = 5.5V			1.0			1.0	mA
I_{OS}	Output short circuit current[2]	V_{CC} = Max, V_{OUT} = 0V	−15		−100	−15		−100	mA
I_{CC}	Supply current	V_{CC} = Max, inputs open		70	110		70	110	mA

NOTES: 1. Commercial—V_{CC}(min) = 4.75V; V_{CC}(max) = 5.25V. Military—V_{CC}(min) = 4.50V; V_{CC}(max) = 5.50V. 2. No more than one output should be shorted at a time.

AC ELECTRICAL CHARACTERISTICS FOR 8X01A V_{CC} = 5V, T_A = +25°C

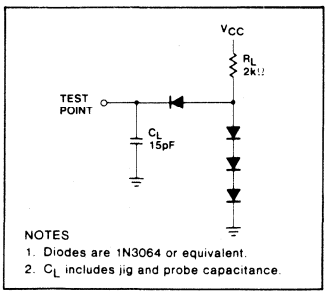
PARAMETER	DESCRIPTION	FROM	TO	TEST CONDITIONS	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
					Min	Typ	Max	Min	Typ	Max	
f_{max}	Max clock freq				12			12			MHz
PULSE WIDTHS:											
t_w-$\overline{CP}$(L)	Clock low			See figure 2	35			35			ns
t_w-$\overline{P}$(L)	Preset low			See figure 3	35			35			ns
t_w-MR(H)	Master reset high			See figure 4	35			35			ns
SETUP/HOLD TIMES:											
t_s-D	Setup time	Data	Clock		55			55			ns
t_s-CWE	Setup time	CWE	Clock	See figure 5	55			55			ns
t_h-D & CWE	Hold time	Data & CWE	Clock		0			0			ns
PROPAGATION DELAY:											
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{PRESET}$	Data output	See figures 1, 2, & 3			55			55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	Master reset	Data output	See figure 4			55			55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{PRESET}$	Error output	See figure 3			55			55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	Master reset	Error output	See figure 4			55			55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{CP}$	Data output	See figure 2			55			55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{CP}$	Error output	See figure 2			55			55	ns
t_{REC}	Recovery time	Preset, MR	Clock	See fig. 3 & 4	35			35			ns

AC ELECTRICAL CHARACTERISTICS FOR 9401 V_{CC} = 5V, T_A = +25°C

PARAMETER	DESCRIPTION	FROM	TO	TEST CONDITIONS	LIMITS (COMMERCIAL)			LIMITS (MILITARY)			UNIT
					Min	Typ	Max	Min	Typ	Max	
f_{max}	Max clock freq				12	20		12	20		MHz
PULSE WIDTHS:											
t_w-$\overline{CP}$(L)	Clock low			See figure 2	35			35			ns
t_w-$\overline{P}$(L)	Preset low			See figure 3	40	30		40	30		ns
t_w-MR(H)	Master reset high			See figure 4	35	25		35	25		ns
SETUP/HOLD TIMES:											
t_s-D	Setup time	Data	Clock		55	35		55	35		ns
t_s-CWE	Setup time	CWE	Clock	See figure 5	55	35		55	35		ns
t_h-D & CWE	Hold time	Data & CWE	Clock		0	−8		0	−8		ns
PROPAGATION DELAY:											
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{PRESET}$	Data output	See figures 1, 2, & 3		40	60		40	60	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	Master reset	Data output	See figure 4		30	55		30	55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{PRESET}$	Error output	See figure 3		40	60		40	60	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	Master reset	Error output	See figure 4		40	60		40	60	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{CP}$	Data output	See figure 2		30	55		30	55	ns
$t_{PLH,PHL}$	Low-to-High and High-to-Low	$\overline{CP}$	Error output	See figure 2		40	60		40	60	ns
t_{REC}	Recovery time	Preset, MR	Clock	See fig. 3 & 4	35	25		35	25		ns

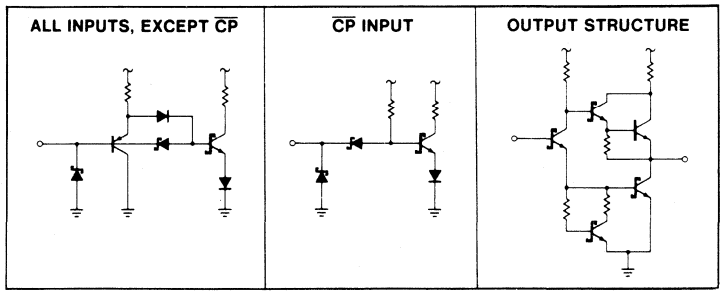
PRELIMINARY

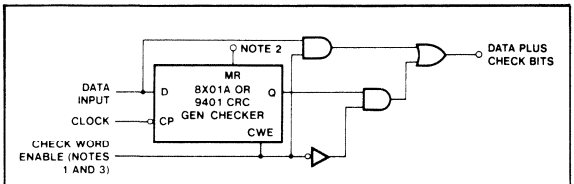
TEST CIRCUIT



NOTES
1. Diodes are 1N3064 or equivalent.
2. C_L includes jig and probe capacitance.

INPUT/OUTPUT STRUCTURES





NOTES
1. Check Word Enable is HIGH while data is being clocked; it is LOW during transmission of check bits.
2. The 8X01A (or 9401) must be RESET or PRESET before each computation.
3. CRC check bits are generated and appended to data bits.

Figure 1. Check Word Generation

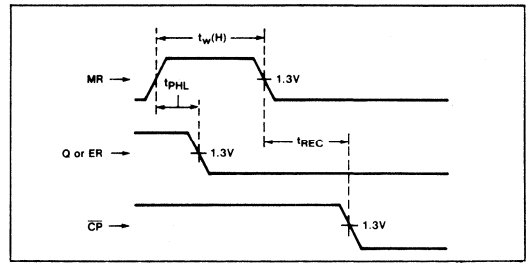


Figure 4. Propagation Delay—MR to Q and ER; Recovery Time—MR to $\overline{CP}$

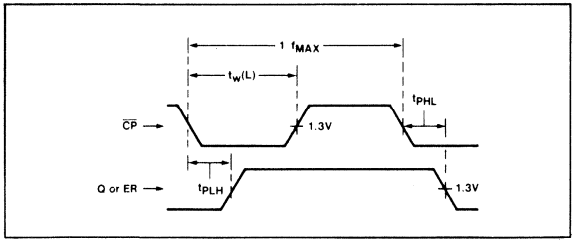


Figure 2. Propagation Delay—$\overline{CP}$ to Q and $\overline{CP}$ to ER

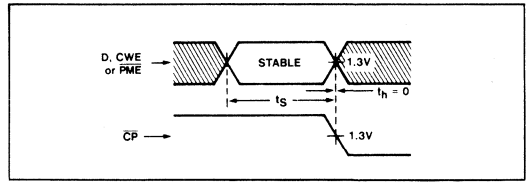


Figure 5. Setup and Hold Times—D to $\overline{CP}$, CWE to $\overline{CP}$, and $\overline{PME}$ to $\overline{CP}$

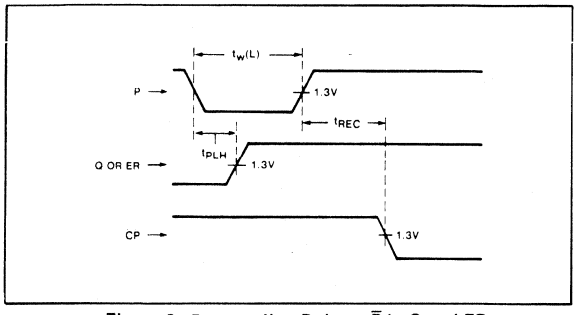


Figure 3. Propagation Delay—$\overline{P}$ to Q and ER; Recovery Time—$\overline{P}$ to $\overline{CP}$

64-BIT FIFO BUFFER MEMORY (16×4)

Originally published by Signetics January 1984

FEATURES

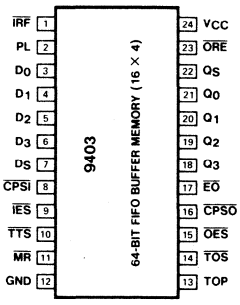
- **10MHz Serial or Parallel Data Rate**
- **Serial or Parallel Input and Output**
- **Expandable Without External Logic**
- **Three-State Outputs**
- **Fully TTL-Compatible**
- **Slim (0.4 in.) 24-Pin DIP**

PRODUCT DESCRIPTION

The 9403 is an expandable fall-through type First-In First-Out (FIFO) Buffer Memory that is optimized for high-speed disc/tape controllers and communication-buffer applications. In multiples of four, the device can be expanded to any number of bits and subsequently, to any number of words. Serial or parallel data can be asynchronously entered or retrieved which makes the 9403 *the* cost-effective solution for implementing buffer memories.

PIN DESIGNATIONS & DESCRIPTIONS

N PACKAGE



TOP VIEW

ORDER NUMBER

N9403N

MNEMONIC AND FUNCTION		DESCRIPTION
$\overline{IRF}$	= Input register full output	Low when input register is full
PL	= Parallel load input	High on PL enables D_0-D_3; not edge-triggered, 1's catching
D_0-D_3	= Parallel data input	—
D_S	= Serial data input	—
$\overline{CPSI}$	= Serial input clock	Edge-triggered and activates on falling edge
$\overline{IES}$	= Serial input enable	When low, serial input is enabled
$\overline{TTS}$	= Transfer to stack input	When low, initiates fall-through
$\overline{MR}$	= Master Reset	Active low
TOP	= Transfer out parallel input	When high and $\overline{TOS}$ is low, enables word transfer from stack to output register—not edge-triggered

MNEMONIC AND FUNCTION		DESCRIPTION
$\overline{TOS}$	= Transfer out serial input	When low and TOP is high, enables word transfer from stack to output register—not edge-triggered
$\overline{OES}$	= Serial output enable input	When low, enables serial output
$\overline{CPSO}$	= Serial output clock input	Edge-triggered and activates on falling edge
$\overline{EO}$	= Output enable	Active low
Q_0-Q_3	= Parallel data output	
Q_S	= Serial data output	
$\overline{ORE}$	= Output register empty output	When high, output register contains valid data
GND	= Ground	—
V_{CC}	= Supply voltage	+5 volts

FUNCTIONAL DESCRIPTION

As shown in Figure 1, the 9403 consists of three parts which operate asynchronously and are virtually independent. These parts are:

- **Input Register**—with serial and parallel data inputs and control signals that permit easy expansion and a handshake interface.
- **FIFO Stack**—4-bit wide, 14-word deep fall-through type with self-contained control logic.
- **Output Register**—with serial and parallel data outputs and control signals that permit easy expansion and a handshake interface.

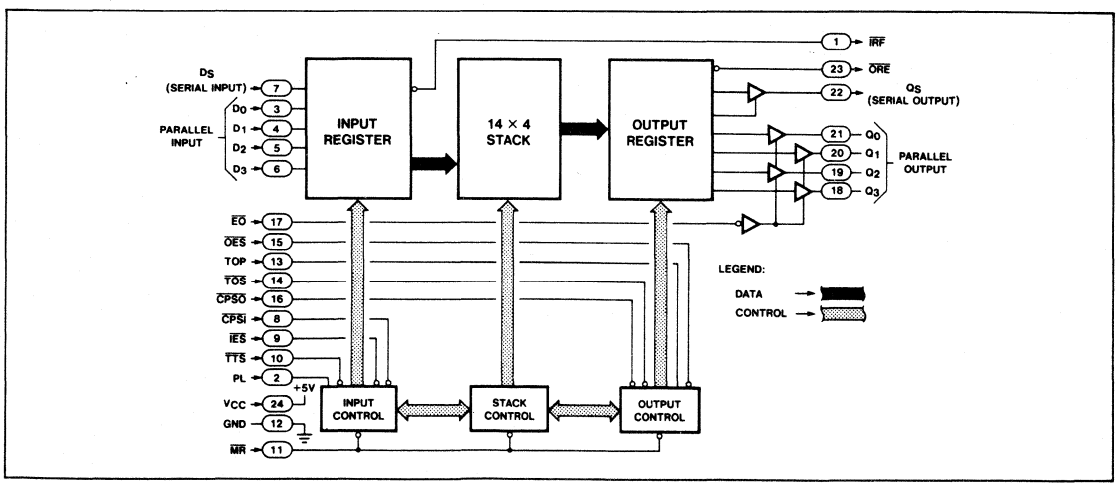


Figure 1. Simplified Block Diagram of 9403 Buffer Memory

INPUT REGISTER

Data can be entered serially or, using the parallel mode of operation, data is entered in 4-bit increments. In either case, the data is subsequently transferred to the fall-through stack; the functional equivalent of this register is shown in Figure 2. The Input Register Full ($\overline{IRF}$) status signal is internally generated by the Register Status (RS) flip-flop; when initialized, the $\overline{Q}$ ($\overline{IRF}$) output of this flip-flop is high.

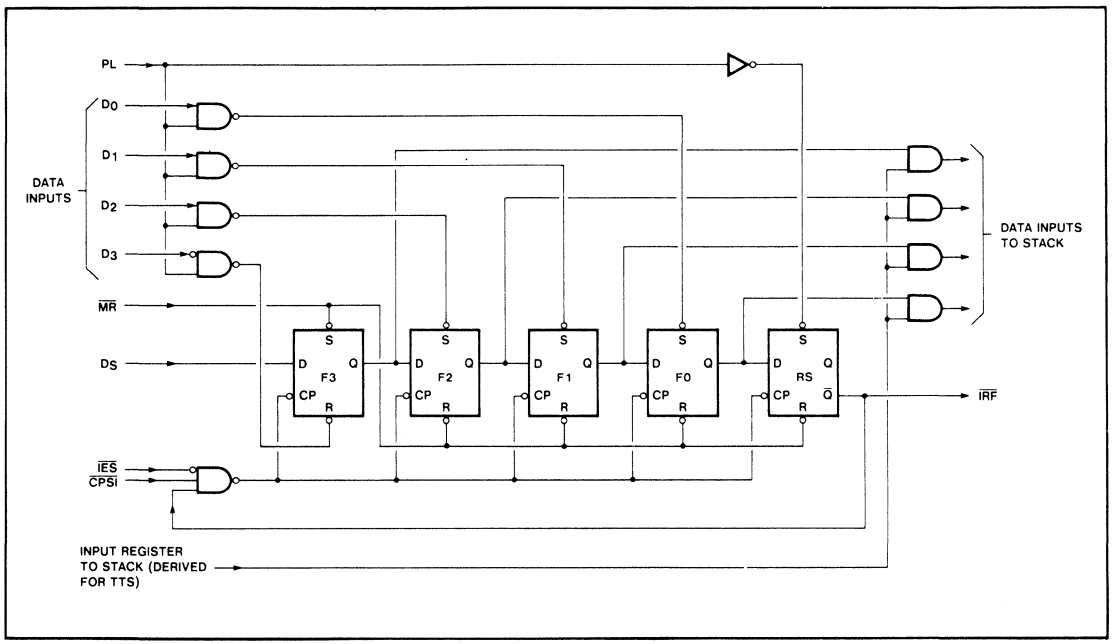


Figure 2. Functional Equivalent of Input Register

Serial Entry (Input Register)

Serial data is entered via the D_S input and is handled by a 5-bit shift register consisting of flip-flops F3, F2, F1, F0, and RS. With $\overline{IES}$ and PL both low, each high-to-low transition of the serial input clock ($\overline{CPSI}$) shifts the input data in domino order from F3 to F2 to F1 to F0. After the fourth clock transition, the four bits of serial data are aligned in F3 through F0 and RS is set, forcing $\overline{IRF}$ low and inhibiting $\overline{CPSI}$ until contents of the input register are transferred to the stack. Figure 3 shows how a serial train of 64-bits would appear in the 9403—four bits (B60-B63) in the input register, 56 bits (B4-B59) in the stack, and four bits (B0-B3) in the output register.

Parallel Entry (Input Register)

When PL is high and $\overline{CPSI}$ is low (Figure 2), flip-flops F0-F3 are loaded with data and $\overline{IRF}$ is forced low. This condition remains until current data is transferred to the stack. Once the data is transferred, $\overline{IRF}$ is driven high and new data can again be clocked into the input flip-flops. If parallel expansion is not being implemented, $\overline{IES}$ must be low to establish row mastership—refer to discussion of parallel expansion.

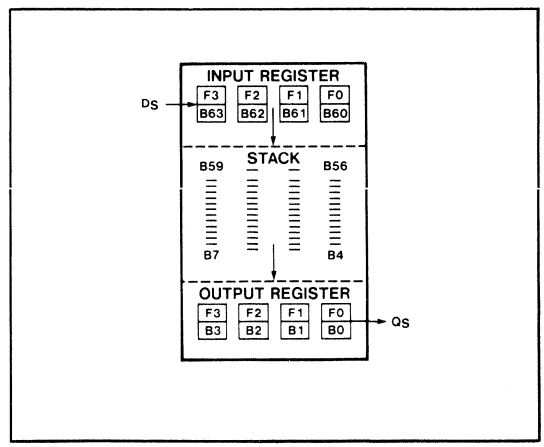


Figure 3. Final Bit Positions Resulting from a Serial Train of 64-Bits

STACK OPERATION

As shown in Figure 2, the outputs of F0-F3 are applied to the stack under control of a signal derived from $\overline{TTS}$. When $\overline{TTS}$ is low, an attempt to inititate a fall-through action is made. If the top location of stack is empty, data is loaded and the input register is re-initialized provided PL is low. Note that initialization is postponed until PL is again low. Thus, automatic FIFO action is achieved by connecting the $\overline{TTS}$ input to the $\overline{IRF}$ output.

The RS flip-flop (Figure 2) records the fact that data has been transferred to the stack; this flip-flop is not cleared until PL goes low. Therefore, if a particular data word is transferred to the stack and falls to the second location before PL goes low, the same word will not be re-transferred even though $\overline{IRF}$ and $\overline{TTS}$ are still low. Once data enters the stack, "fall-through" is automatic; a delay is necessary only when waiting for the next stack location to empty. In the 9403, as in most modern FIFO designs, the $\overline{MR}$ input initializes the stack control section and does not clear the data.

OUTPUT REGISTER

This register receives and stores 4-bits of data from the bottom stack location and, on demand, outputs the data onto a three-state 4-bit parallel data bus or a three-state serial data bus. The Output Register Full ($\overline{ORE}$) status signal is internally-generated by the FX flip-flop; when data is transferred from the stack to the output register, $\overline{ORE}$ goes high. The functional equivalent of the output register is shown in Figure 4.

Retrieval of Parallel Data

With the stack empty and $\overline{MR}$ in the active-low state, the $\overline{ORE}$ output goes low, signifying that the output register is also empty. When new data is entered and has fallen through to bottom location of the stack, it is automatically transferred to the output register, provided the Transfer Out Parallel (TOP) input is high. When the data is transferred from stack-to-register, $\overline{ORE}$ goes high and valid data appears at Q_0-Q_3 (Figure 4), provided the three-state buffers are enabled, that is, $\overline{EO}$ is active-low. When TOP goes low, $\overline{ORE}$ is driven low which indicates that the data output cycle is complete; however, the original data remains latched in the flip-flops until the next word (if available) is transferred from the stack to the output register.

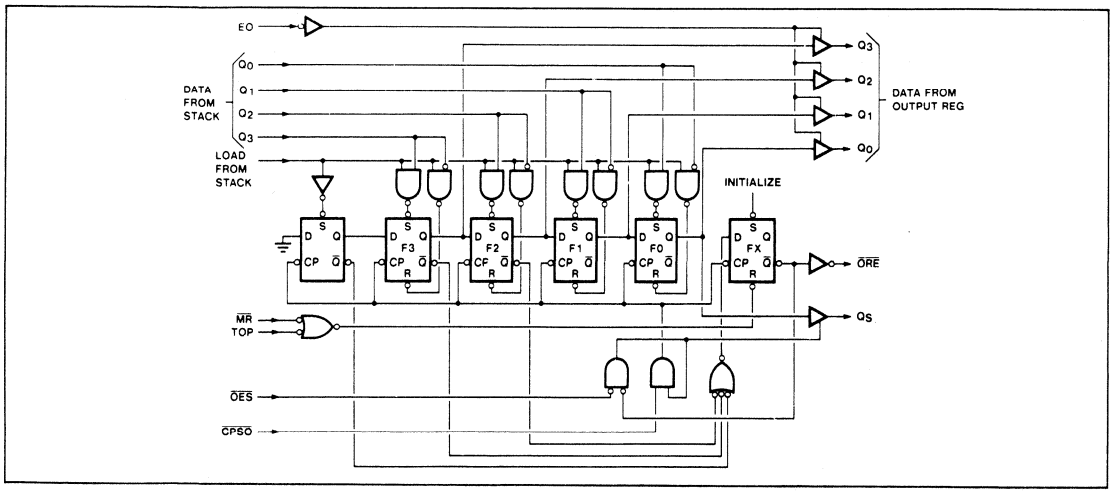


Figure 4. Functional Equivalent of Output Register

Retrieval of Serial Data

When the FIFO stack is empty and $\overline{MR}$ is driven low, the $\overline{ORE}$ output goes low to indicate that the output register is ready to accept new data from the stack. After new data is entered and falls through to the bottom stack location, it is transferred to the output register provided $\overline{TOS}$ is low and TOP is high. As a result of the data transfer, $\overline{ORE}$ goes high indicating valid data in the output register. Subsequently, the $\overline{Q}_S$ output is automatically enabled and the first data bit is transmitted to the three-state serial data bus. Henceforth, a serial shift of data occurs on each high-to-low transition of $\overline{CPSO}$. On the fourth transition, the register is emptied, $\overline{ORE}$ is forced low, and serial output $\overline{Q}_S$ is disabled. To request a new word from the stack, the $\overline{TOS}$ input can be connected to the $\overline{ORE}$ output.

For parallel operation, $\overline{CPSO}$ must be low, whereas, $\overline{TOS}$ should be grounded for single-slice operation or connected to the appropriate $\overline{ORE}$ for expanded operation. The TOP input is not edge-triggered; therefore, if it goes high before data is available from stack but data becomes available before it goes low, the data will be transferred to the output register. However, internal control circuits prevent the same data from being transferred twice. If TOP goes high and returns to low before data is available from the stack, $\overline{ORE}$ will remain low, indicating the absence of valid output data.

VERTICAL EXPANSION

In a vertical structure, the 9403 can be expanded to achieve greater word capacity without any external parts; a 46-word by 4-bit FIFO is shown in Figure 5. Using the same technique and similar connections, any FIFO of 15n+1 words (where *n* is the number of devices) can be constructed. Observe that word expansion does not sacrifice flexibility of the 9403 FIFO as regards serial/parallel input and output.

HORIZONTAL EXPANSION

The 9403 can be horizontally expanded to store long words in multiples of 4-bits, again without external logic. Connections required to form a 16-word by 12-bit FIFO are shown in Figure 6; using similar techniques, any 16-word by 4n-bit FIFO (where *n* is the number of devices) can be constructed.

For horizontal or bit expansion, it is good practice to connect, respectively, the $\overline{IRF}$ and $\overline{ORE}$ outputs of the right-most device (most significant device) to the $\overline{TTS}$ and $\overline{TOS}$ inputs of all devices to the left (least significant devices) to guarantee that no operation is initiated before each and every device is ready. Word expansion does not affect the ability of the 9403 to handle serial/parallel inputs and outputs; however, the ripple form of expansion shown in Figure 6 does extract a penalty in speed of operation. Whereas a single 9403 is guaranteed to operate at 10MHz, an array of four FIFOs connected as shown is guaranteed to operate at 4.3MHz.

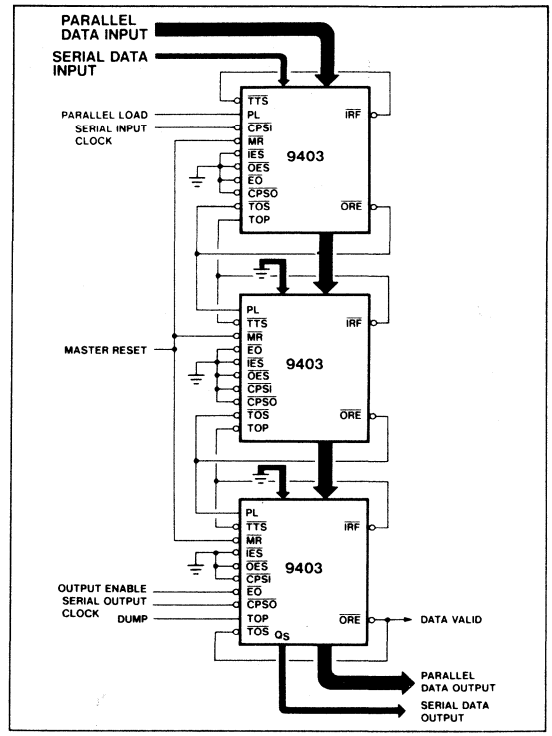


Figure 5. Word Expansion

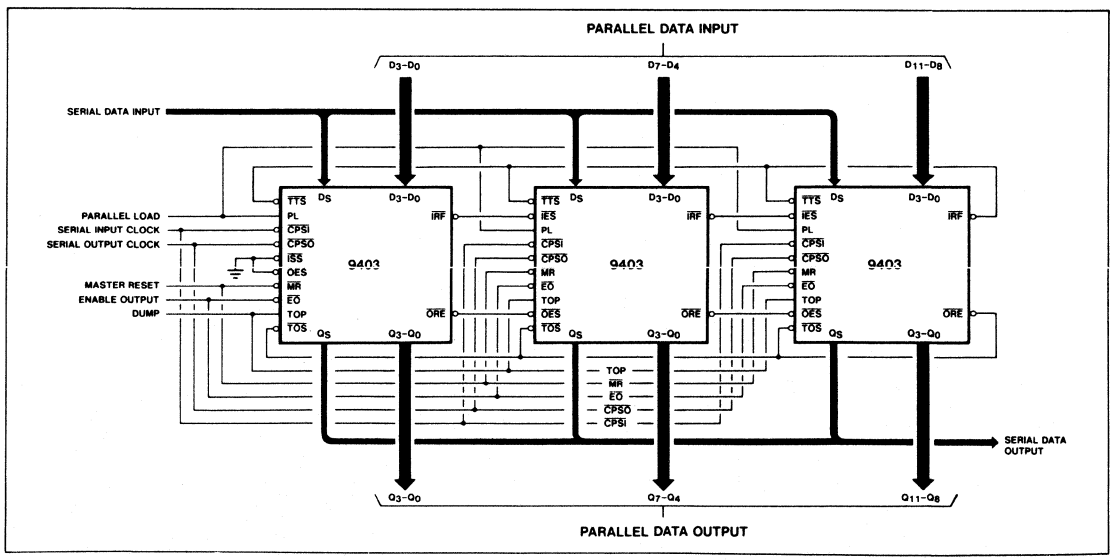
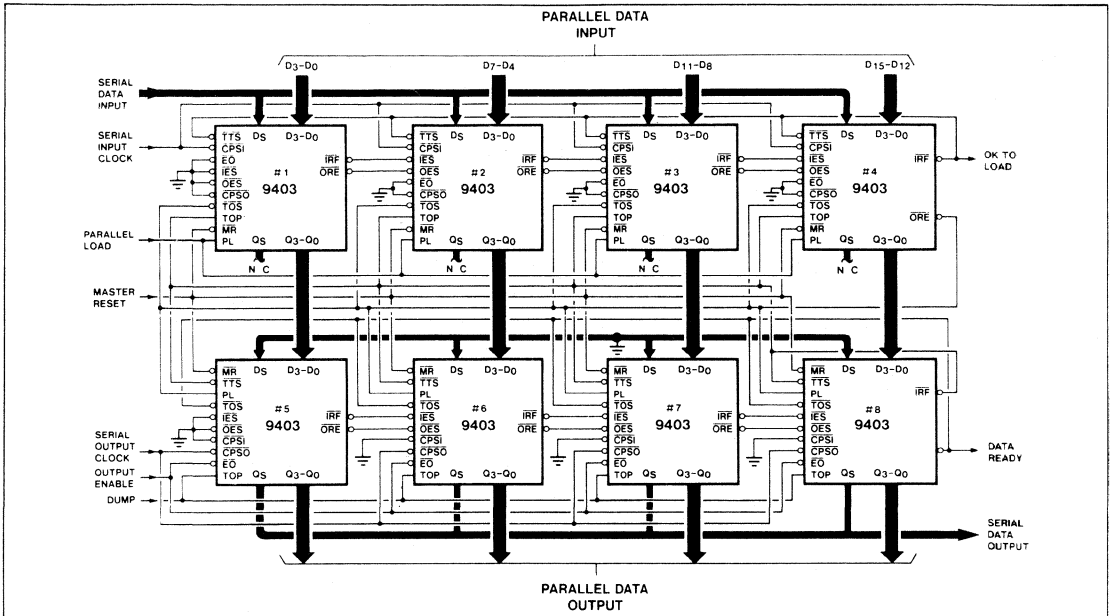


Figure 6. Bit Expansion

HORIZONTAL AND VERTICAL EXPANSION

In addition to bit-or-word expansion, the 9403 can be used to expand in both the horizontal and vertical directions; a 31-word by 16-bit FIFO is shown in Figure 7. Using the same or similar techniques, any FIFO of 15m+1 words by 4n-bits can be constructed, where *m* is the number of devices in a column and *n* is the number of devices in a row.

The chart appended to Figure 7 shows the final positions for a contiguous serial entry of 496 bits. Figures 8 and 9, respectively, show the timing relationships involved for data-entry and data-retrieval pertaining to the 31-word by 16-bit array.



INPUT REGISTER	UNIT NUMBER & ORDER OF SERIAL BITS #1	#2	#3	#4	OUTPUT REGISTER	UNIT NUMBER & ORDER OF SERIAL BITS #5	#6	#7	#8
D_3	BIT 483	BIT 487	BIT 491	BIT 495	Q_3	BIT 3	BIT 7	BIT 11	BIT 15
D_2	482	486	490	494	Q_2	2	6	10	14
D_1	481	485	489	493	Q_1	1	5	9	13
D_0	480	484	488	492	Q_0	0	4	8	12

Figure 7. Horizontal and Vertical Expansion—31X16 FIFO

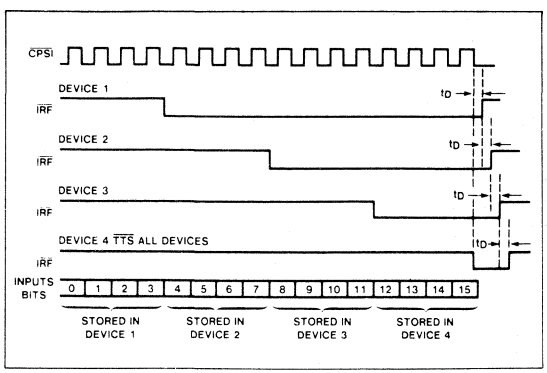


Figure 8. Entry of Serial Data for Array of Figure 7

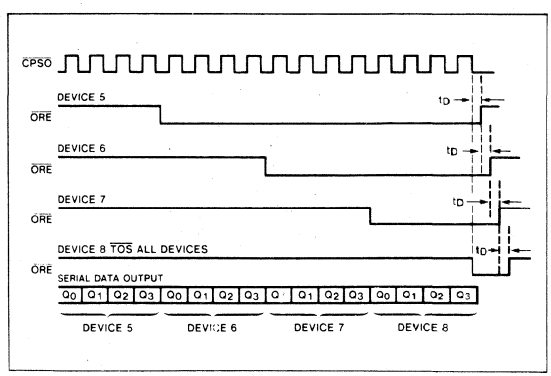


Figure 9. Retrieval of Serial Data for Array of Figure 7

INTERLOCKING CIRCUITS

Most conventional FIFO designs provide the status-signal counterparts of $\overline{IRF}$ and $\overline{ORE}$. However, when these devices are used in arrays, variations in unit-to-unit operating speeds require the use of external gating to ensure that all devices have, in fact, completed the last operation. The 9403 incorporates simple but effective master/slave interlocking circuits to eliminate these gating requirements.

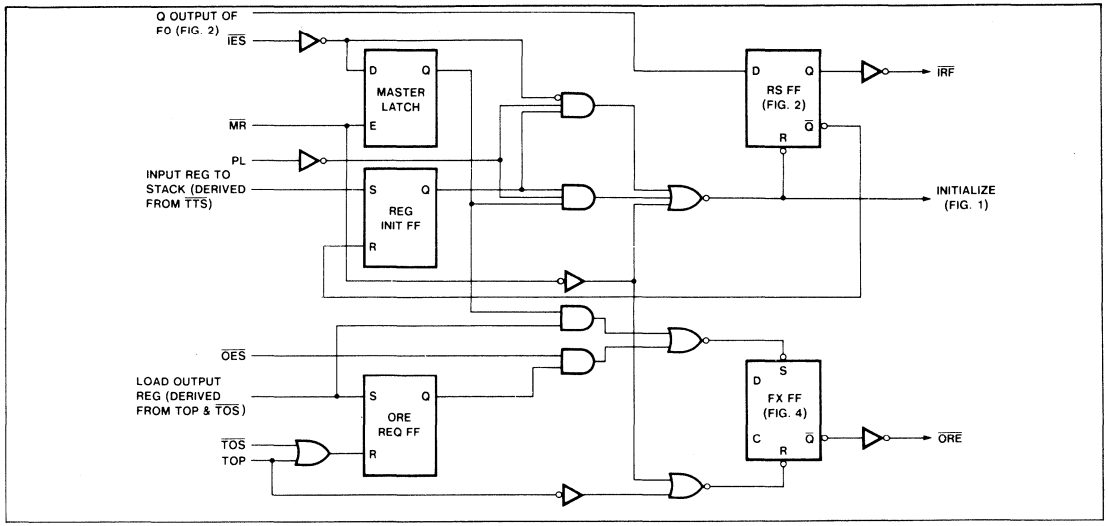


Figure 10. Functional Equivalent of Interlocking Circuits

ABSOLUTE MAXIMUM RATINGS

PARAMETER		RATING	UNIT
V_{CC}	Power supply voltage	+7	Vdc
V_{IN}	Input voltage	+5.5	Vdc
V_O	Off-state output voltage	+5.5	Vdc
T_A	Operating temperature range	0 to +70	°C
T_{stg}	Storage temperature range	−65 to +150	°C

DC ELECTRICAL CHARACTERISTICS Over operating temperature range unless otherwise noted

PARAMETER		TEST CONDITIONS[1,2]	LIMITS			UNIT
			Min	Typ	Max	
V_{IH}	Input high voltage	Guaranteed input high voltage	2.0			V
V_{IL}	Input low voltage	Guaranteed input low voltage			0.8	V
V_{CD}	Input Clamp Diode Voltage	V_{CC} = Min, I_{IN} = −18mA		−0.9	−1.5	V
V_{OH}	Output high voltage, $\overline{ORE}$, $\overline{IRF}$	V_{CC} = Min, I_{OH} = −400μA	2.4	3.4		V
V_{OH}	Output high voltage, Q_0-Q_3, Q_S	I_{OH} = −5.7mA, V_{CC} = Min	2.4	3.1		V
V_{OL}	Output low voltage, Q_0-Q_3, Q_S	V_{CC} = Min, I_{OL} = 16mA		0.35	0.5	V
V_{OL}	Output low voltage, $\overline{ORE}$, $\overline{IRF}$	V_{CC} = Min, I_{OL} = 8.0mA		0.35	0.5	V
I_{OZH}	Output off current high, Q_0-Q_3, Q_S	V_{CC} = Max, V_{OUT} = 2.4V, V_E = 2V			100	μA
I_{OZL}	Output off current low, Q_0-Q_3, Q_S	V_{CC} = Max, V_{OUT} = 0.5V, V_E = 2V			−100	μA
I_{IH}	Input high current	V_{CC} = Max, V_{IN} = 2.7V		1.0	40	μA
		V_{CC} = Max, V_{IN} = 5.5V			1.0	mA
I_{IL}	Input low current, all except $\overline{OES}$ & $\overline{IES}$				−0.36	mA
		V_{CC} = Max, V_{IN} = 0.4V			−0.96	
I_{OS}	Output short circuit current, Q_0-Q_3, Q_S $\overline{ORE}$, $\overline{OES}$	V_{CC} = Max, V_{OUT} = 0, (Note 3)	−30		−130	mA
I_{CC}	Supply Current	V_{CC} Max, Inputs open		115	170	mA

NOTES
1. Operating temperature ranges are guaranteed after terminal equilibrium has been reached.
2. All voltages measured with respect to ground terminal.
3. No more than one output should be shorted at a time.

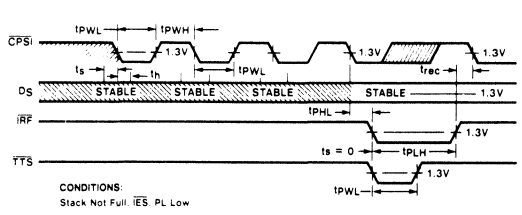
AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V, C_L = 15pF, T_A = 25° C

PARAMETER	FROM INPUT	TO OUTPUT	TEST CONDITIONS[1,2,3]	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT
FALL-THROUGH TIME: t_{DFT}	Positive going PL	Q_0-Q_3	$\overline{TTS}$ connected to $\overline{IRF}$, $\overline{TOS}$ connected to $\overline{ORE}$, IES, $\overline{OES}$, EO, $\overline{CPSO}$ low, TOP high (f, Fig. 11)		450	600	ns
PROPAGATION DELAY:							ns
t_{PLH} Low-to-high	Negative going $\overline{TTS}$	$\overline{IRF}$	Stack not full, PL low (a & b, Fig. 11)		48	64	
t_{PHL} High-to-low	Negative going $\overline{CPSI}$	$\overline{IRF}$			18	25	
t_{PLH} Low-to-high	Negative going $\overline{CPSO}$	Q_S	Serial output $\overline{OES}$ low, TOP high (c & d, Fig. 11)		30	40	ns
t_{PHL} High-to-low					17	28	
t_{PHL} High-to-low	Negative going $\overline{CPSO}$	$\overline{ORE}$			32	42	ns
t_{PLH} Low-to-high	Positive going TOP	Q_0-Q_3	$\overline{EO}$, $\overline{CPSO}$ low (e, Fig 11)		40	56	ns
t_{PHL} High-to-low					31	45	
t_{PLH} Low-to-high	Positive going TOP	$\overline{ORE}$	Parallel output, $\overline{EO}$, $\overline{CPSO}$ low (e, Fig. 11)		51	68	ns
t_{PHL} High-to-low	Negative going TOP	$\overline{ORE}$			40	54	
t_{PLH} Low-to-high	Negative going $\overline{TOS}$	Positive going $\overline{ORE}$	Data in stack, TOP high, (c & d, Fig. 11)		41	56	ns
t_{PHL} High-to-low	Positive going PL	Negative going $\overline{IRF}$	Stack not full (g & h, Fig. 11)		20	33	ns
t_{PLH} Low-to-high	Negative going PL	Positive going $\overline{IRF}$			33	46	ns
t_{PLH} Low-to-high	Positive going $\overline{OES}$	$\overline{ORE}$			26	44	
t_{PLH} Low-to-high	Positive going IES	Positive going $\overline{IRF}$			31	40	
ENABLE DELAY:							ns
t_{PZH} High	$\overline{EO}$	Q_0-Q_3	Out of high impedance state		9	14	
t_{PZL} Low						20	
t_{PZL} Low	Negative going $\overline{OES}$	Q_S			13	25	ns
t_{PZH} High						20	
DISABLE DELAY:							ns
t_{PLZ} Low	$\overline{EO}$	Q_0-Q_3	Into high impedance state		7	14	
t_{PHZ} High							
t_{PLZ} Low	Negative going $\overline{OES}$	Q_S			7	14	ns
t_{PHZ} High							
APPEARANCE TIME:							ns
t_{AP} Parallel	$\overline{ORE}$	Q_0-Q_3	Time elapsed between $\overline{ORE}$ going high and valid data appearing at output, negative number indicates data available before $\overline{ORE}$ goes high		−12	−5	
t_{AS} Serial	$\overline{ORE}$	Q_S			6	10	
PULSE WIDTH							ns
t_{PWL} $\overline{CPSI}$ low			Stack not full, PL low (a & b, Fig. 11)	20	11		
t_{PWH} $\overline{CPSI}$ high				33	19		
t_{PWL} TOP low			$\overline{CPSO}$ low, data available in stack (e, Fig. 11)	30	17		ns
t_{PWH} TOP high				26	13		
t_{PWL} $\overline{CPSO}$ low			TOP high, data in stack, (c & d, Fig. 11)	30	16		ns
t_{PWH} $\overline{CPSO}$ high				32	18		
t_{PWH} PL high			Stack not full (g & h, Fig. 11)	40	29		ns
t_{PWL} $\overline{TTS}$ low (serial or parallel mode)			Stack not full (a, b, g, & h, Fig. 11)	20	9		ns
t_{PWL} $\overline{MR}$ low			(f, Fig. 11)	25	13		ns
SETUP and HOLD TIME:							ns
t_s Setup time	D_S	Negative $\overline{CPSI}$	PL low (a & b, Fig. 11)	28	17		
t_h Hold time	D_S	$\overline{CPSI}$		0	−6		

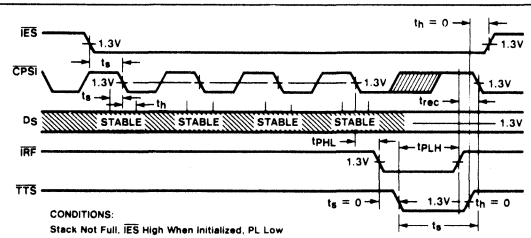
AC ELECTRICAL CHARACTERISTICS V_{CC} = 5.0V, C_L = 15pF, T_A = 25° C (Cont'd)

PARAMETER		FROM	TO	TEST CONDITIONS[1,2,3]	LIMITS			UNIT
					Min	Typ	Max	
t_s	Setup time	Parallel inputs	PL	Length of time parallel inputs must be applied prior to rising edge of PL	0	−22		ns
t_h	Hold time	Parallel inputs	PL	Length of time parallel inputs must remain applied after falling edge of PL	2			ns
t_s	Set up time (serial or parallel mode)	$\overline{TTS}$	$\overline{IRF}$	(a, b, g, & h, Fig. 11)	0	−20		ns
t_s	Setup time	Negative going $\overline{ORE}$	Negative going $\overline{TOS}$	TOP high (c & d, Fig. 11)	0	−24		ns
t_s t_s	Setup time Setup time	Negative going $\overline{IES}$ Negative $\overline{TTS}$	$\overline{CPSI}$ $\overline{CPSI}$	(b, Fig. 11)	45 84	23 58		ns
RECOVERY TIME: t_{rec}		$\overline{MR}$	Any input	(f, Fig. 11)	15	5		ns

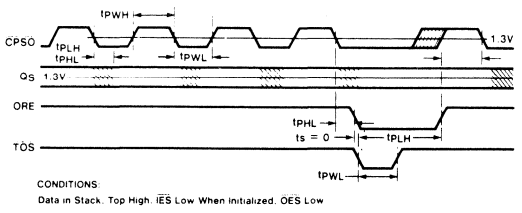
NOTES
1. Initialization requires a master reset to occur after power has been applied
2. $\overline{TTS}$ normally connected to $\overline{IRF}$
3. If stack is full, $\overline{IRF}$ will stay low



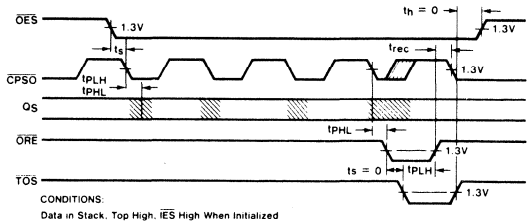
a. Serial Input, Unexpanded or Master Operation



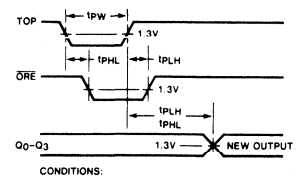
b. Serial Input, Expanded Slave Operation



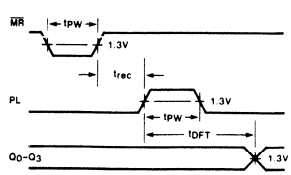
c. Serial Output, Unexpanded or Master Operation



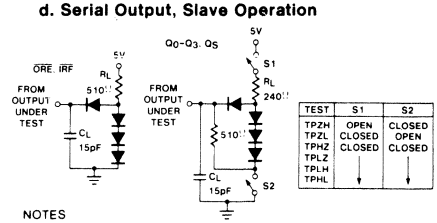
d. Serial Output, Slave Operation



e. Parallel Output, 4-Bit Word or Master/Slave Operation in Parallel Expansion



f. Fall-Through Time



NOTES
1. C_L includes jig and probe capacitance
2. All diodes are 1N3064 or equivalent

g. Test Circuit for Measurement of AC Parameters

Figure 11. 9403 Timing and Parameter-Measurement Information

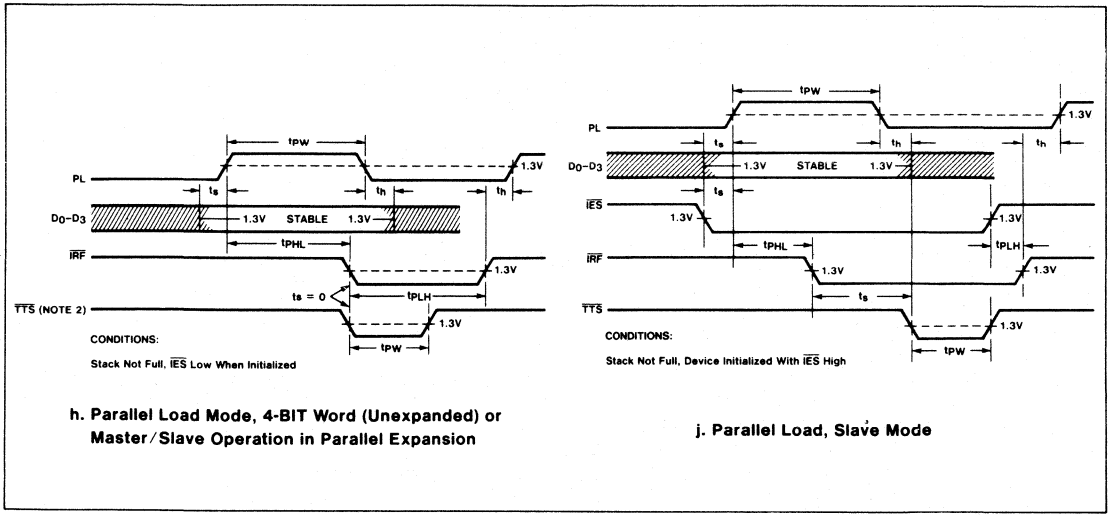
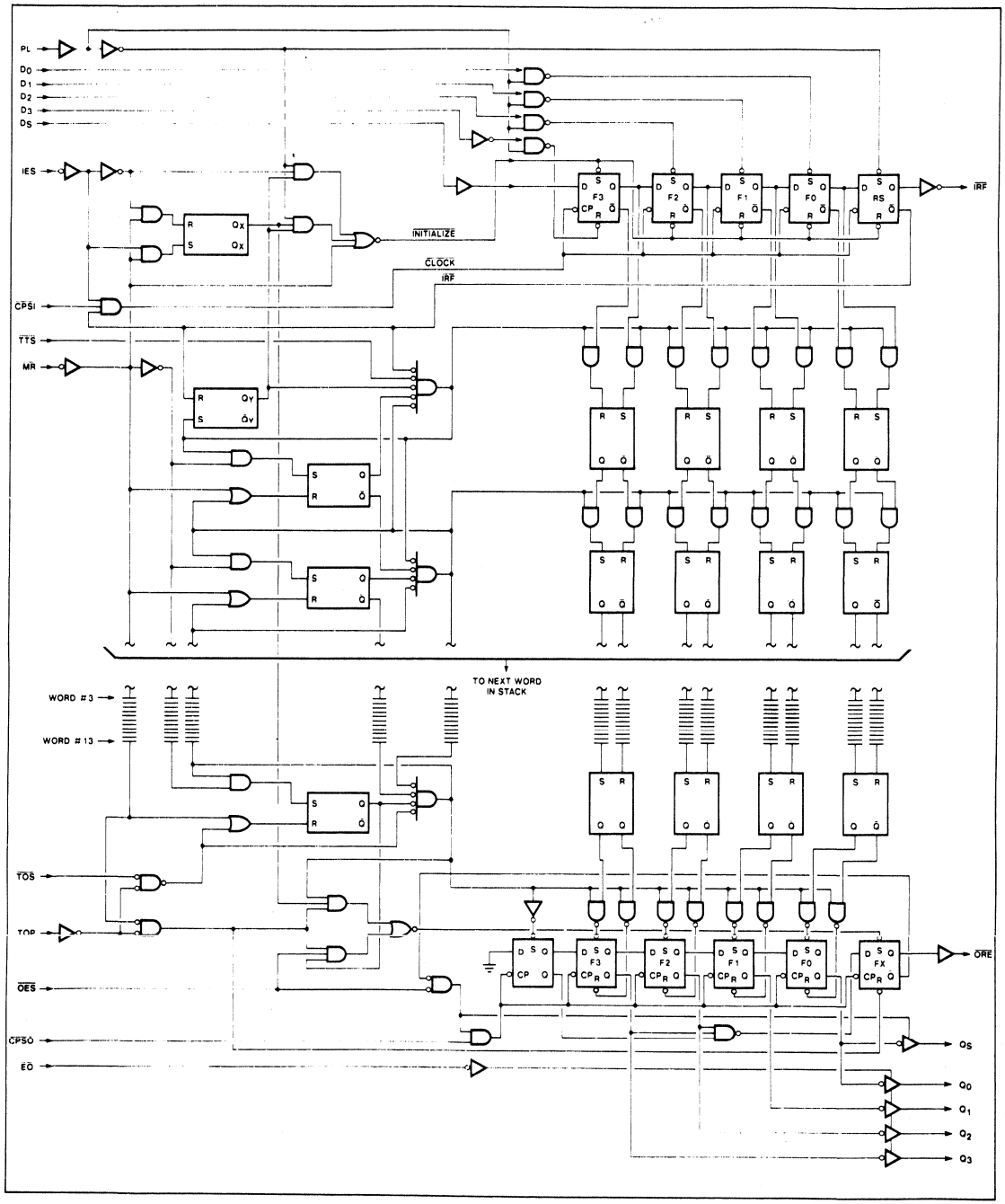


Figure 11. 9403 Timing and Parameter-Measurement Information (Cont'd)

LOGIC DIAGRAM



AUTODIRECTIONAL BUS TRANSCEIVER

Originally published by Signetics January 1984

DESIGN FEATURES

- **Intelligent bidirectional bus repeater with self-generating or external control**
- **Eight independent channels**
- **Open-collector outputs (meets DEC UNIBUS* specifications)**
- **TTL compatible**
- **High speed (30-nanoseconds max)**
- **Expandable to any number of bits**
- **High input impedance for every operating value of V_{CC}**
- **Low input current (less than 100-microamperes); high output current (up to 70-milliamperes)**
- **0.6 in. 24-pin DIP**
- **+5V supply**

USE AND APPLICATION

- **Minicomputers**
- **Microcomputers MOS/Bipolar**
- **Communications**
- **Signal buffer**
- **Bus fan-out extensions**
- **Distributed processing**
- **Bidirectional bus connector/isolator**

PRODUCT DESCRIPTION

The Signetics 8X41 Autodirectional Bus Transceiver is a general purpose asynchronous device ideal for system bus expansion applications. The 8X41 consists of eight data channels, each with one pair of terminals (A_i and B_i); each data channel can be operated independently.

The device requires no external controls since all intelligence is internally generated; thus, operation of the device is completely autonomous. The first logic low signal that occurs on one channel terminal (A_i or B_i) will be repeated on the corresponding terminal (B_i or A_i) of the same channel.

The 8X41 is designed for use in open-collector bus systems where high speed and low-current inputs/high-current outputs are required. In system configurations, the discrete capabilities of the bus transceiver can be expanded by parallel connection to service any number of bits. To provide reliable operation and integrity of data transfers, all channels are disabled by an on-chip power monitor whenever V_{CC} falls below approximately 4V.

FUNCTIONAL OPERATION

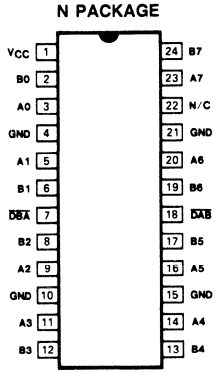
The 8X41 (Figure 1) consists of eight functionally independent yet logically identical channels. Each channel consists of two bus terminals (A_i and B_i); each terminal is internally connected to an open-collector driver and a high-impedance receiver. The monitoring state of each channel is defined when both terminals (A_i and B_i) are "high"; in this state, the internal logic of the 8X41 continually examines the A and B bus signals to determine signal direction—A_i to B_i or B_i to A_i. A low signal occurring at either of the two terminals causes the open-collector driver on the opposite terminal to follow suit; hence, the signal is repeated by the 8X41. For each channel, latches L1 and L2 determine signal direction. As shown in the truth table for these latches, there is no transmission of data when both signals are low; however, this condition should never occur during normal system operation.

The internal automatic direction control can be overridden by either or both of the common disable inputs—$\overline{DBA}$ and $\overline{DAB}$. When $\overline{DBA}$ is driven low ($\overline{DAB}$ = high), the B_i to A_i path is interrupted and the device becomes a unidirectional repeater in the A_i to B_i direction only. With these conditions reversed ($\overline{DAB}$ = low and $\overline{DBA}$ = high), the A_i to B_i path is interrupted and the chip functions as a unidirectional repeater in the B_i to A_i direction. When both control signals are low, data passage is inhibited in both directions. Refer to the I/O truth table for all possible input/output conditions.

*Trademark of the Digital Equipment Corporation

8X41 PACKAGE/PIN DESIGNATIONS

N PACKAGE



TOP VIEW

ORDER NUMBER

N8X41N

PIN NO.	IDENTIFIER	FUNCTION
1	V_{CC}	+5V power
3, 5, 9, 11, 14, 16, 20, and 23	A_0-A_7	A-bus input/output
2, 6, 8, 12, 13, 17, 19, and 24	B_0-B_7	B-bus input/output
4, 10, 15, and 21	GND	Circuit ground
7 & 18	$\overline{DBA}$ & $\overline{DAB}$	Input/output control
22	N/C	No connection

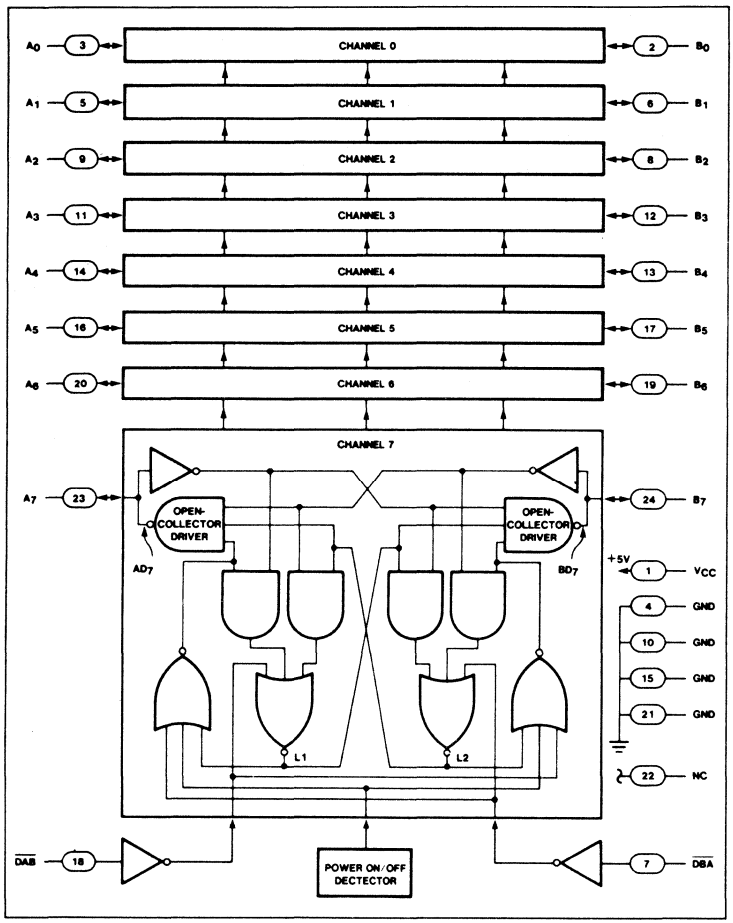


Figure 1. Logic Diagram of 8X41

$\overline{DBA}$	$\overline{DAB}$	FUNCTION
Ø	Ø	Data transmission inhibited
Ø	1	$A_i \rightarrow B_i$
1	Ø	$A_i \leftarrow B_i$
1	1	$A_i \rightarrow B_i$ $A_i \leftarrow B_i$

i = Channel 0, 1, 2, 3, 4, 5, 6, or 7
$A_i \rightarrow B_i$ = Data transmission from A_i to B_i
$A_i \leftarrow B_i$ = Data transmission from B_i to A_i

TRUTH TABLE FOR INTERNAL LATCHES

LATCHES		DIRECTION OF DATA
L1	L2	
1	1	Monitoring state
1	Ø	A_i to B_i
Ø	1	B_i to A_i
Ø	Ø	No transmission

INPUT/OUTPUT TRUTH TABLE

EXTERNAL CONTROLS		INPUT SIGNALS		OUTPUT DRIVER SIGNALS	
$\overline{DAB}$	$\overline{DBA}$	A_i	B_i	AD_i	BD_i
H	H	L	L	H	H
H	H	L	H	H	L
H	H	H	L	L	H
H	H	H	H	H	H
H	L	L	L	H	L
H	L	L	H	H	L
H	L	H	L	H	H
H	L	H	H	H	H
L	H	L	L	L	H
L	H	L	H	H	H
L	H	H	L	L	H
L	H	H	H	H	H
L	L	X	X	H	H

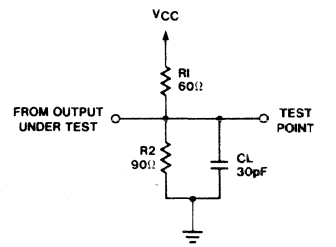
Notes:
A_i = External signal
AD_i = Output A driver
B_i = External signal
BD_i = Output B driver
X = Don't care

DC CHARACTERISTICS $V_{CC} = 5V (\pm 5\%)$; $T_A = 0°C$ to $70°C$

PARAMETER	DESCRIPTION	TEST CONDITIONS	LIMITS			UNITS
			Min	Typ	Max	
V_{OL}	Bus output low voltage (driver ON)	I_{OL} = 70 mA; V_{CC} = Min			0.5	V
*V_B	Bus input threshold voltage (driver OFF)		1.3		1.7	V
V_{IH} ($\overline{DBA}$, $\overline{DAB}$ only)	High level input voltage		2.0			V
V_{IL} ($\overline{DBA}$, $\overline{DAB}$ only)	Low level input voltage				0.8	V
V_{IC}	Input clamp voltage	V_{CC} = Min; I_{IL} = −18mA			−1.5	V
V_{PD}	Power ON/OFF detector threshold voltage		3.7		4.35	V
I_{IH} ($\overline{DBA}$, $\overline{DAB}$ only)	High level input current	V_{CC} = Max; V_{IN} = 2.7V			20	μA
I_{IL} ($\overline{DBA}$, $\overline{DAB}$ only)	Low level input current	V_{CC} = Max; V_{IN} = 0.4V			−0.4	mA
I_I	Bus input current (driver OFF)	V_{CC} = Max; V_B = 2.5V*			100	μA
		V_{CC} = Max; V_B = 0V*			−20	μA
I_{OFF}	Bus leakage current (power OFF)	V_{CC} = 0V; V_B = 2.5V*			100	μA
I_{CC}	Supply current	V_{CC} = Max; A_0-A_7 = Low or B_0-B_7 = Low and $\overline{DBA}$ = $\overline{DAB}$ = High		145	180	mA

*$V_B = V_{BUS}$

LOAD CIRCUIT FOR OUTPUTS



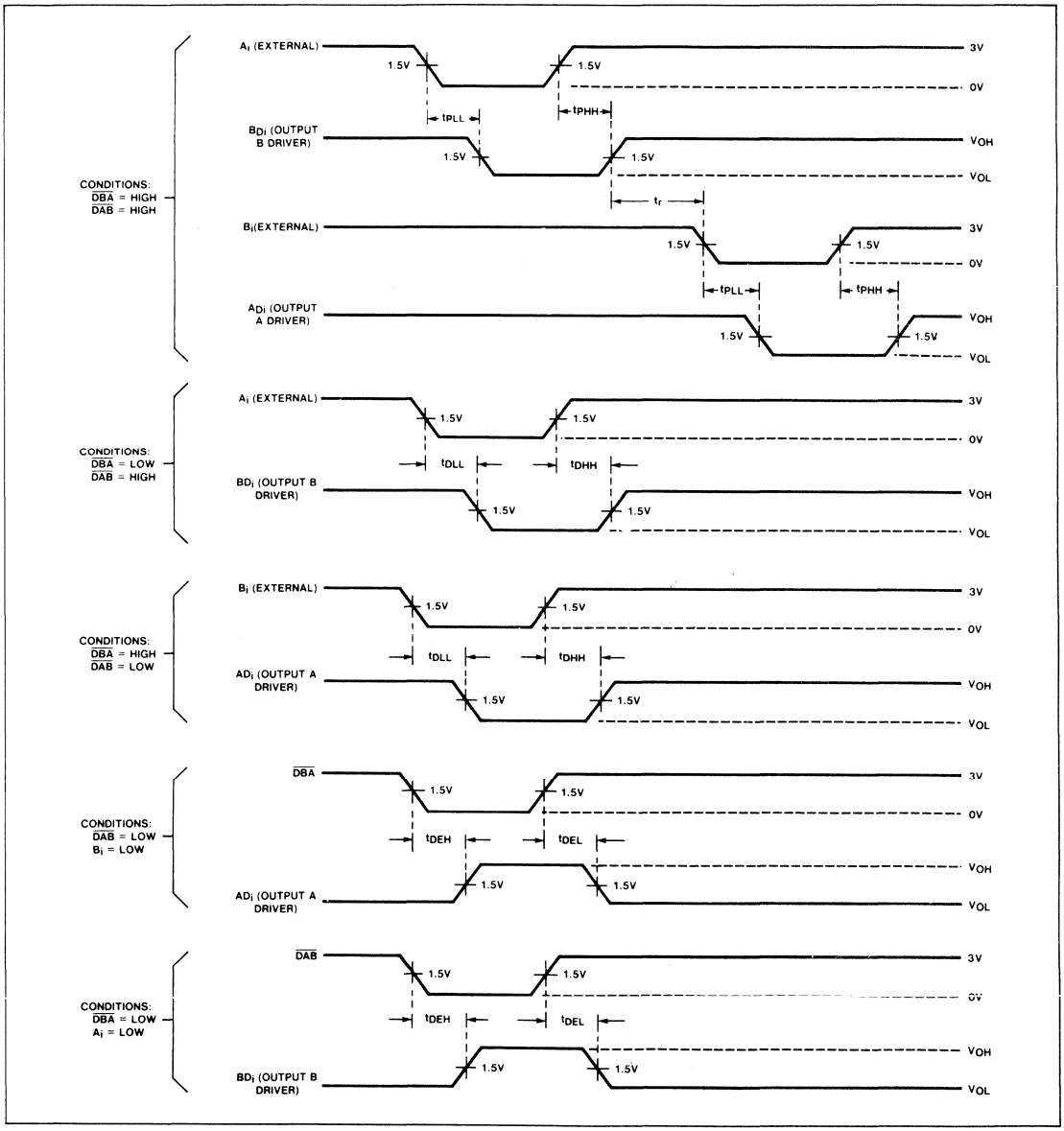
Note:
C_L includes probe and jig capacitance

AC CHARACTERISTICS $V_{CC} = 5V (\pm 5\%)$; $T_A = 0°C$ to $70°C$

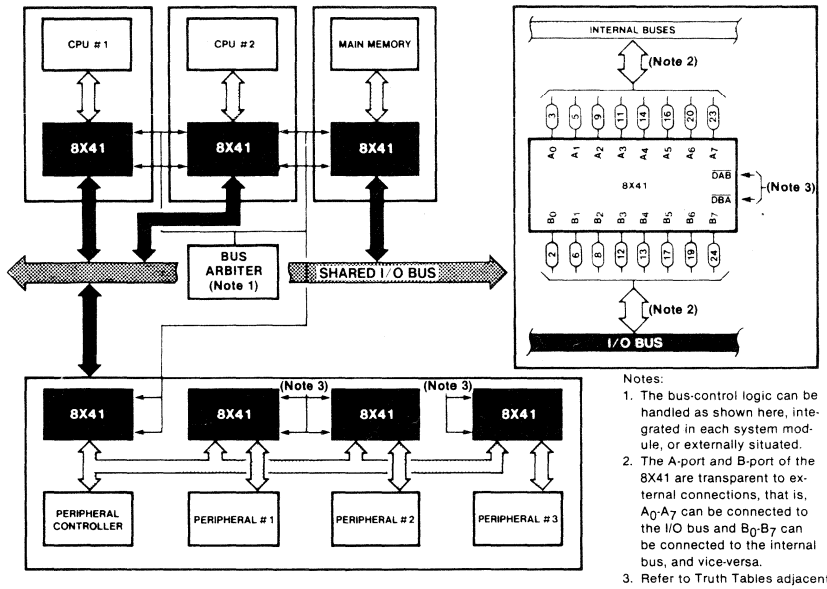
PARAMETER	DESCRIPTION	FROM	TO	TEST CONDITIONS	LIMITS			UNITS
					Min	Typ	Max	
t_{PLL}	Propagation delay	Low A_i Low B_i	Low BD_i Low AD_i	$\overline{DBA}$ = $\overline{DAB}$ = High			30	ns
t_{PHH}	Propagation delay	High A_i High B_i	High BD_i High AD_i	$\overline{DBA}$ = $\overline{DAB}$ = High			30	ns
t_{DHH}	Propagation delay	High A_i	High BD_i	$\overline{DBA}$ = Low; $\overline{DAB}$ = High			25	ns
		High B_i	High AD_i	$\overline{DAB}$ = Low; $\overline{DBA}$ = High			25	ns
t_{DLL}	Propagation delay	Low A_i	Low BD_i	$\overline{DBA}$ = Low; $\overline{DAB}$ = High			25	ns
		Low B_i	Low AD_i	$\overline{DAB}$ = Low; $\overline{DBA}$ = High			25	ns
t_{DEH}	Propagation delay	Low $\overline{DBA}$	High AD_i	$\overline{DAB}$ = Low; B_i = Low			30	ns
t_{DEL}	Propagation delay	High $\overline{DBA}$	Low AD_i	$\overline{DAB}$ = Low; B_i = Low			30	ns
t_{DEH}	Propagation delay	Low $\overline{DAB}$	High BD_i	$\overline{DBA}$ = Low; A_i = Low			30	ns
t_{DEL}	Propagation delay	High $\overline{DAB}$	Low BD_i	$\overline{DBA}$ = Low; A_i = Low			30	ns
t_r	Recovery time (see timing diagram)	—	—	$\overline{DBA}$ = $\overline{DAB}$ = High		20		ns

Notes: A_i = External signal AD_i = Output A driver B_i = External signal BD_i = Output B driver

8X41 TIMING DIAGRAM



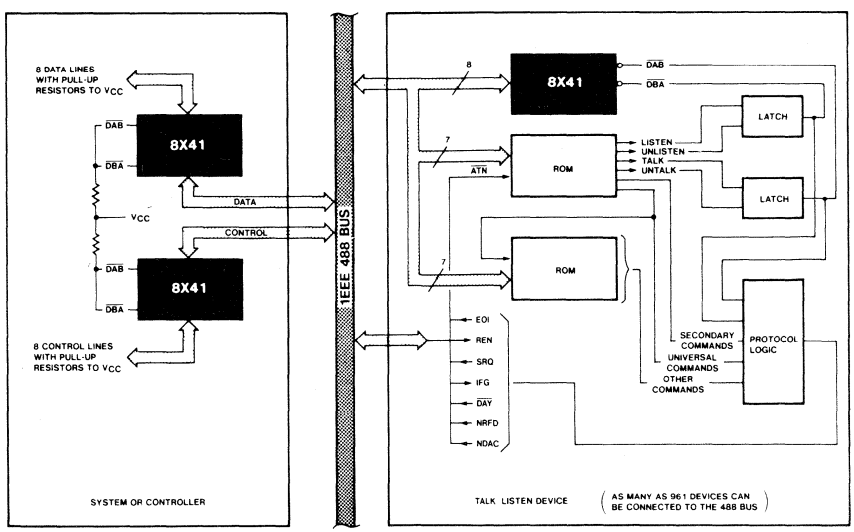
USING THE 8X41 IN A BUS-SHARED CONFIGURATION



Notes:

1. The bus-control logic can be handled as shown here, integrated in each system module, or externally situated.
2. The A-port and B-port of the 8X41 are transparent to external connections, that is, A_0-A_7 can be connected to the I/O bus and B_0-B_7 can be connected to the internal bus, and vice-versa.
3. Refer to Truth Tables adjacent to Figure 1 for data control capabilities.

INTERFACING 8X41 TO IEEE 488 BUS



FIFO RAM CONTROLLER (FRC)

Originally published by Signetics January 1984

DESIGN FEATURES

- **12-Bit FIFO Address Generator**
- **Data Rate Exceeding 8MHz**
- **Asynchronous Read/Write Operations**
- **Three-State Address Outputs**
- **User-Defined Word Width**
- **Specifically Designed for Use with High-Speed Bipolar RAMs (Adaptable for Use with MOS RAMs)**
- **TTL Input and Output**
- **16mA Address-Drive Capability**

USE AND APPLICATION

- **Interface Between Independently-Clocked Systems**
- **Buffer Memories for Disk and/or Tape**
- **Data Communication Concentrators**
- **CPU/Terminal Buffering**
- **DMA Applications**
- **CRT Terminals**

PRODUCT DESCRIPTION

The Signetics 8X60 FIFO RAM Controller (FRC) is an address and status generator designed to implement a high-speed/high-capacity First-In/First-Out (FIFO) stack utilizing standard off-the-shelf RAMs—see **APPLICATIONS** on the last page of this data sheet. The FRC can control up to 4096 words of buffer memory; intermediate buffer sizes can be selected—refer to the memory length table on the next page. Built-in arbitration logic handles read/write operations on a first-come/first-served basis.

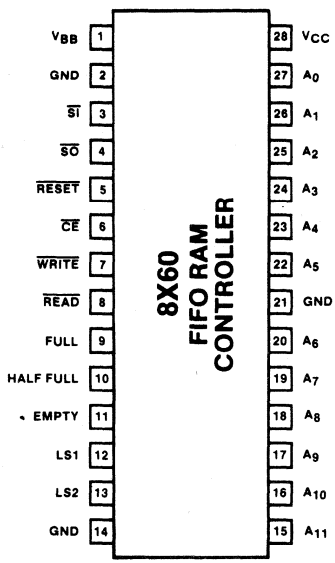
As shown in Figure 1, the FRC consists of:

- A 12-Bit Write Address Generation Counter (Counter #1) and a 12-Bit Read Address Generation Counter (Counter #2).
- A 12-Bit Up/Down Status Counter (Counter #3).
- Twelve Three-State Address Drivers.
- Control Logic.

The two address counters, #1 and #2, respectively, are used to generate write and read addresses; the outputs of these counters are multiplexed to the three-state address drivers. Counter #3 generates *full*, *empty*, and *half full* status.

PACKAGE AND PIN DESIGNATIONS

N, FQ PACKAGE



TOP VIEW

ORDER NUMBERS

N8X60N, RB8X60F

PIN NO.	IDENTIFIER	FUNCTION
1	V_{BB}	Supply voltage for internal circuits.
2, 14, 21	GND	Circuit ground.
3	$\overline{SI}$	Shift-In request for write cycle; active-low input.
4	$\overline{SO}$	Shift-Out request for read cycle; active-low input.
5	$\overline{RESET}$	Active-low master reset input.
6	$\overline{CE}$	Active-low chip enable input.
7	$\overline{WRITE}$	Write cycle address valid; active-low output.
8	$\overline{READ}$	Read cycle address valid; active-low output.
9	FULL	Memory full status output; also, override input capability. Active when *high*.
10	HALF FULL	Memory half-full status output; active-high.
11	EMPTY	Memory empty status output; also, override input capability. Active when *high*
12	LS1	Least significant bit (LSB) of the memory length select input.
13	LS2	Most significant bit (MSB) of the memory length select input.
15-20 22-27	A_{11}-A_0	Three-state address outputs; A_0 = LSB.
28	V_{CC}	Supply voltage.

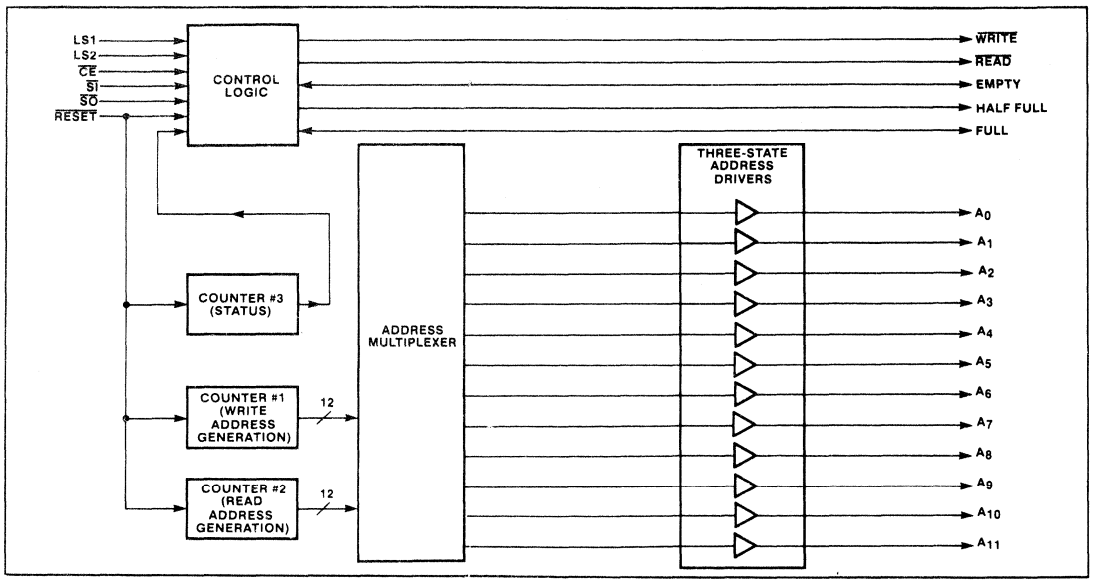


Figure 1. Functional Block Diagram of FIFO RAM Controller

FUNCTIONAL OPERATION

The FRC operates in either of two basic modes—*write* into the FIFO buffer memory or *read* from the FIFO buffer memory. These two operations are described in subsequent paragraphs and the complete sequence is summarized in Table 1. Typical Write/Read timing relationships, arbitration logic, and chip-enable control are shown in the **Timing Diagrams.**

FIFO BUFFER MEMORY—WRITE CYCLE

To perform a write operation, $\overline{SO}$ must be *high* and SI must be *low*. When these conditions exist and other control parameters (Table 1) are satisfied, the write address in Counter #1 (Figure 1) is output to the address bus via the multiplexer and $\overline{WRITE}$ output goes *low*. (*Note.* Normally, the $\overline{WRITE}$ output goes *low* after the address output becomes stable—refer to *WRITE CYCLE TIMING DIAGRAM*. The $\overline{WRITE}$ output may then act as a *write* or *chip* enable for the RAMs that are used to implement the memory.)

When the *write* cycle is ended ($\overline{SI}$ is forced high), the $\overline{WRITE}$ output goes *high*, the address output buffers return to a high-impedance state, Counter #1 (Write Address Generation) and Counter #3 (Status) are both incremented, and Counter #2 (Read Address Generation) remains unchanged.

FIFO BUFFER MEMORY—READ CYCLE

To perform a read operation, $\overline{SI}$ must be *high* and $\overline{SO}$ must be *low*. When these conditions exist and other control parameters (Table 1) are satisfied, the read address contained in Counter #2 (Figure 1) is output to the address bus and the $\overline{READ}$ output goes *low*.

When the *read* cycle is ended ($\overline{SO}$ is forced *high*), the $\overline{READ}$ output goes *high*, the output buffers return to a high-impedance state, Counter #2 (Read Address Generation) is incremented, Counter #3 (Status) is decremented, and Counter #1 (Write Address Generation) remains unchanged.

CONTROL LOGIC

To prevent the possibility of operational conflicts. $\overline{SI}$ and $\overline{SO}$ are treated on a first-come/first-served basis; these two input signals are controlled by internal arbitration logic—refer to the applicable **TIMING DIAGRAMS** and **AC CHARACTERISTICS** for functional and timing relationships. If one cycle is requested while the other cycle is in progress, the requested cycle will commence as soon as the current-cycle is complete (provided other control parameters are satisfied).

As shown in the accompanying diagram, the buffer length of the FIFO memory can be hardware-selected via the Length Select (LS1, LS2) inputs. When less than the maximum length is selected, the unused high-order bits of the address outputs are held in the high-impedance state.

MEMORY LENGTH

LS1	LS2	HALF LENGTH	FULL LENGTH
L	L	2048	4096
H	L	32	64
L	H	512	1024
H	H	128	256

Generation of the status output signals (HALF FULL, FULL and EMPTY) is a function of the Length Select (LS1, LS2) inputs and the current state of Status Counter #3. In general, the status outputs reflect the conditions that follow:

- **HALF FULL**—this status output signals goes *high* on the positive-going edge of $\overline{SI}$ if the MSB of the selected length of Counter #3 becomes a "1". The HALF FULL signal will go from *high-to-low* on the positive-going edge of $\overline{SO}$ when, after the *read* cycle, the selected length of Counter #3 changes from "100...00" to "011...11". For example, if the selected memory length is 256 words (FULL = 256), then HALF FULL = 128 words; hence, on the

positive-going edge of $\overline{SO}$ when Counter #3 reaches a count of 127, the HALF FULL output will go from *high*-to-*low*.

- **FULL**—this signal serves both as a status output and as an override input. The FULL signal goes *high* on the negative-going edge of $\overline{SI}$ if all bits of Counter #3 for selected length are equal to "1". The FULL output goes from *high*-to-*low* on the negative-going edge of $\overline{SO}$.
- **EMPTY**—this signal also serves as a status output and as an override input. On the negative-going edge of $\overline{SO}$, the EMPTY output is driven *high* if Status Counter #3 contains a value of "1"; on the positive-going edge of $\overline{SO}$, the counter is decremented to "0". The EMPTY output goes from *high*-to-*low* on the negative-going edge of $\overline{SI}$.

Once the FULL signal is *high*, further Write Cycle Requests ($\overline{SI}$ = low) are ignored; similarly, once the EMPTY signal is *high*, further Read Cycle requests ($\overline{SO}$ = low) are ignored. However, to accommodate diversified applications, the FULL and EMPTY outputs are open-collector with on-chip 4.7K passive pull-up resistors. If either the FULL or EMPTY pins are forced *low* via external control, the corresponding *write* or *read* cycle may resume (provided the external FULL or EMPTY input is held *low* until the corresponding $\overline{WRITE}$ or $\overline{READ}$ output goes *low*) and the address/status counters will continue normal operation*—refer to Table 1.

The user must force the $\overline{RESET}$ input *low* to initialize the chip. (*Note.* If the $\overline{RESET}$ signal is driven *low* during a *write* or *read* cycle, the address output may have a short period of uncertainty before assuming a high-impedance state.) The following actions occur when $\overline{RESET}$ is active:

- All internal counters are set to "0".
- All address output lines are forced to the high-impedance state.
- HALF FULL and FULL outputs are forced *low*.
- $\overline{WRITE}$, $\overline{READ}$, and EMPTY outputs are forced high.

When $\overline{CE}$ is *high*, the address output lines are forced to the high-impedance state, further *write* or *read* cycle requests are ignored, and all counters remain unchanged. If $\overline{CE}$ switches from *low*-to-*high* during a *write* or *read* cycle, the cycle in progress is always completed before the disabled state is entered. For details of these operations, refer to the timing information shown later in this data sheet.

*Refer to **Note** on inside back cover

Table 1. Summary of Operation

INPUTS				INITIAL	RESULTING OUTPUTS			
$\overline{RESET}$	$\overline{CE}$	$\overline{SI}$	$\overline{SO}$	CONDITIONS	$\overline{WRITE}$	$\overline{READ}$	ADDRESS BUS	COMMENTS
L	X	X	X		H	H	Hi-Z	Reset all counters to 0.
H	X	H	H		H	H	Hi-Z	No action
H	L	L	H	FULL = L	L	H	Write address from Ctr #1	Shift into FIFO stack (Write Cycle)
H	L	L	H	FULL = H	H	H	Hi-Z	Stack full (Write inhibited)
H	L	H	L	EMPTY = L	H	L	Read address from Ctr #2	Shift out of FIFO stack (Read Cycle)
H	L	H	L	EMPTY = H	H	H	Hi-Z	Stack empty (Read inhibited)
H	L	L	↓	Write cycle in progress	L	H	Write address from Ctr #1	Continue write cycle (until $\overline{SI}$ goes high)
H	L	↓	L	Read cycle in progress	H	L	Read address from Ctr #2	Continue read cycle (until $\overline{SO}$ goes high)
H	L	L	L	EMPTY = H	L	H	Write address from Ctr #1	Shift in (Read inhibited)
H	L	L	L	FULL = H	H	L	Read address from Ctr #2	Shift out (Write inhibited)
H	L	↑	H	Write cycle in progress	↑	H	Goes to Hi-Z	Increment write address counter #1 and status counter #3
H	L	H	↑	Read cycle in progress	H	↑	Goes to Hi-Z	Increment read address counter #2; decrement status counter #3
H	L	↑	L	Write cycle in progress (Note 1)	↑	↓	Changes to read address from Ctr #2	Increment write address counter #1 and status counter #3
H	L	L	↑	Read cycle in progress (Note 2)	↓	↑	Changes to write address from Ctr #1	Increment read address counter #2; decrement status counter #3
H	H	↓	H		H	H	Hi-Z	Chip disabled
H	H	H	↓		H	H	Hi-Z	Chip disabled
H	↑	L	X	FULL = L; write cycle begun (Note 1)	L	H	Write address from Ctr #1	Continue write cycle (until $\overline{SI}$ goes high)
H	↑	X	L	EMPTY = L; read cycle begun (Note 2)	H	L	Read address from Ctr #2	Continue read cycle (until $\overline{SO}$ goes high)
H	↓	L	L	FULL = L; EMPTY = L	—	—	—	This set of conditions should be avoided

NOTES

1. Write cycle will occur if either $\overline{SI}$ goes *low* before $\overline{SO}$ goes *low* or EMPTY = H when $\overline{SO}$ goes *low*.
2. Read cycle will occur if either $\overline{SO}$ goes *low* before $\overline{SI}$ goes *low* or FULL = H when $\overline{SI}$ goes *low*.

ABSOLUTE MAXIMUM RATINGS

PARAMETER	DESCRIPTION	RATING	UNIT
V_{CC}	Power Supply Voltage	+7	Vdc
V_{BB}	Supply Voltage for Internal Circuits	+4	Vdc
V_{IN}	Input Voltage	+5.5	Vdc
V_O	Off-State Output Voltage	+5.5	Vdc
T_{STG}	Storage Temperature Range	−65 to +150	°C

CONDITIONS: Commercial— $V_{CC} = 5.0V (\pm 5\%)$, $V_{BB} = 1.5V (\pm 5\%)$[1], $0°C \leq T_A \leq 70°C$
Military— $V_{CC} = 5.0V (\pm 10\%)$, $V_{BB} = 1.5V (\pm 10\%)$[1], $T_A \leq -55°C$, $T_C \leq 125°C$

DC ELECTRICAL CHARACTERISTICS

PARAMETER		TEST CONDITIONS	LIMITS (COMMERCIAL) MIN	TYP[2]	MAX	LIMITS (MILITARY) MIN	TYP[2]	MAX	UNITS
V_{IH}	High level input voltage	Note 3	2.0			2.0			V
V_{IL}	Low level input voltage				0.8			0.8	V
V_{OH}	High level output voltage: All outputs except FULL and EMPTY	V_{CC} = Min; I_{OH} = −2.6mA	2.7	3.5		2.5	3.5		V
V_{OL}	Low level output voltage: Address Bus, $\overline{WRITE}$, $\overline{READ}$	V_{CC} = Min, I_{OL} = 16mA		0.38	0.5		0.38	0.5	V
	HALF FULL, FULL, and EMPTY	V_{CC} = Min; I_{OL} = 8mA		0.35	0.5		0.35	0.5	V
V_{CD}	Diode clamp voltage: All inputs except FULL and EMPTY	V_{CC} = Min; I_{CD} = −18mA	−1.5	−0.8		−1.5	−0.8		V
I_{IH}	High level input current: All inptus except FULL and EMPTY	V_{CC} = Max; V_{IH} = 2.7V		0.1	20		0.1	20	μA
	FULL and EMPTY	V_{CC} = Max; V_{IH} = 2.7V; Stack FULL or Stack EMPTY (Note 3)		−470	−750		−470	−900	μA
I_{IL}	Low level input current: All inputs except FULL and EMPTY	V_{CC} = Max; V_{IL} = 0.4V		−0.17	−0.4		−0.17	−0.4	mA
	FULL and EMPTY	V_{CC} = Max; V_{IL} = 0.4V; Stack FULL or Stack EMPTY		−1.12	−1.8		−1.12	−1.8	mA
I_{OH}	High level output current: FULL, EMPTY	V_{CC} = Min, V_{OH} = V_{CC} (min)		15	100		15	100	μA
I_{OZH}	High-Z output current (HIGH); Address Bus (Three-State)	V_{CC} = Max; V_{OUT} = 2.4V		0.9	20		0.9	20	μA
I_{OZL}	High-Z output current (LOW); Address Bus (Three-State)	V_{CC} = Max; V_{OUT} = 0.5V		−0.6	−20		−0.6	−20	μA
I_I	Input leakage current: All inputs except FULL and EMPTY	V_{CC} = Max; V_{IN} = 5.5V		0.03	0.1		0.03	0.1	mA
I_{OS}	Short-circuit output current: Address Bus and HALF FULL	V_{CC} = Max; V_{OH} = 0V	−15	−68	−100	−15	−68	−100	mA
	$\overline{WRITE}$, $\overline{READ}$	V_{CC} = Max; V_{OH} = 0V	−40	−73	−100	−40	−73	−100	mA
I_{CC}	Supply current from V_{CC}	V_{CC} = Max; Address Bus = High-Z; 0°C →		81	140	−55°C →		140	mA
		70°C →		81	110	125°C →		100	
I_{BB}	Supply current from V_{BB}	V_{BB} = Max; 0°C →		63	95	−55°C	63	100	mA
		70°C →		63	85	125°C	63	90	

NOTES
1. V_{BB} can be obtained from a regulated 1.5V supply; alternately, proper supply current (I_{BB}) can be obtained by connecting a 56-ohm (±5%, 0.5W) resistor in series with V_{CC} as shown later in the **APPLICATIONS** diagram.
2. Typical limits are: V_{CC} = 5.0V; T_A = 25°C.
3. Because of the internal pull-up resistor on the FULL and EMPTY pins, a negative current is required to force the required voltage.
4. V_{OL} at I_{OL} = 4mA for Military part.

CONDITIONS: Commercial— V_{CC} = 5.0V (±5%), V_{BB} = 1.5V (±5%), 0°C ≤ T_A ≤ 70°C; Military— V_{CC} = 5.0V (±10%), V_{BB} = 1.5V (±10%), T_A ≤ −55°C, T_C ≤ 125°C; Loading— See TEST LOADING CIRCUITS

AC ELECTRICAL CHARACTERISTICS

PARAMETERS	REFERENCES FROM	REFERENCES TO	TEST CONDITIONS	LIMITS (Commercial) Min	Typ	Max	LIMITS (Military) Min	Typ	Max	UNITS
PULSE WIDTHS										
T_{LH} $\overline{SI}$ high	↑$\overline{SI}$	↓$\overline{SI}$	Stack approaching FULL (Note 1)	25	13		25	13		ns
T_{DH} $\overline{SO}$ high	↑$\overline{SO}$	↓$\overline{SO}$	Stack approaching EMPTY (Note 1)	30	16		30	16		ns
WRITE CYCLE TIMING										
T_{LA} Address stable delay	↓$\overline{SI}$	An	FULL = Low; $\overline{SO}$ = High		40	55		40	60	ns
T_{AW} Address lead time	An	↓$\overline{WRITE}$		3			0			ns
T_{LAW} $\overline{WRITE}$ output active delay	↓$\overline{SI}$	↓$\overline{WRITE}$	FULL = Low; $\overline{SO}$ = High	35	51	65	35	51	70	ns
T_{LW} $\overline{WRITE}$ output inactive dalay	↑$\overline{SI}$	↑$\overline{WRITE}$			3	10		3	10	ns
T_{WA} Address lag time	↑$\overline{WRITE}$	An		20	34		10	34		ns
T_{LT} Address output disable	↑$\overline{SI}$	An(Hi-Z)			37	60		37	60	ns
T_{LF} FULL status active delay	↓$\overline{SI}$	↑FULL	Stack approaching FULL; $\overline{SO}$ = High		39	65		39	70	ns
T_{LE} EMPTY status inactive delay	↓$\overline{SI}$	↓EMPTY	Stack = EMPTY		40	65		40	65	ns
T_{HFH} HALF-FULL status active delay	↑$\overline{SI}$	↑HALF FULL	Stack approaching HALF-FULL		30	45		30	50	ns
T_{DW} $\overline{WRITE}$ output active after read	↑$\overline{SO}$	↓$\overline{WRITE}$	Both $\overline{SI}$ & $\overline{READ}$ = Low		74	95		74	100	ns
READ CYCLE TIMING										
T_{DA} Address stable delay	↓$\overline{SO}$	An	EMPTY = Low; $\overline{SI}$ = High		40	55		40	60	ns
T_{AR} Address lead time	An	↓READ		−1			−5			ns
T_{DAR} $\overline{READ}$ output active delay	↓$\overline{SO}$	↓READ	EMPTY = Low; $\overline{SI}$ = High	30	48	65		35	70	ns
T_{DR} $\overline{READ}$ output inactive delay	↑$\overline{SO}$	↑$\overline{READ}$			5	10		5	10	ns
T_{RA} Address lag time	↑$\overline{READ}$	An		20	32		10	32		ns
T_{DT} Address output disable	↑$\overline{SO}$	An (Hi-Z)			37	60		37	60	ns
T_{DE} EMPTY status active delay	↓$\overline{SO}$	↑EMPTY	Stack approaching EMPTY; $\overline{SI}$ = High		38	50		38	50	ns
T_{DF} FULL status inactive delay	↓$\overline{SO}$	↓FULL	Stack = FULL		38	50		38	65	ns
T_{HFL} HALF-FULL status inactive delay	↑$\overline{SO}$	↓HALF FULL	Stack exactly HALF-FULL		54	75		54	85	ns
T_{LR} $\overline{READ}$ output active after wirte	↑$\overline{SI}$	↓$\overline{READ}$	Both $\overline{SO}$ & $\overline{WRITE}$ = Low		70	90		70	100	ns
CHIP ENABLE TIMING (WIRTE)										
T_{HEW} Chip enable hold time[2]	↓$\overline{SI}$	↑$\overline{CE}$	FULL = Low; $\overline{SO}$ = High	10	1		10	1		ns
T_{SEW} Chip disable setup time[3]	↑$\overline{CE}$	↓$\overline{SI}$	FULL = Low; $\overline{SO}$ = High	10	1		10	1		ns
T_{PEW} Chip enable delay time	↓$\overline{CE}$	↓$\overline{WRITE}$	FULL = Low; $\overline{SI}$ = Low; $\overline{SO}$ = High		69	95		69	110	ns
CHIP ENABLE TIMING (READ)										
T_{HER} Chip enable hold time[2]	↓$\overline{SO}$	↑$\overline{CE}$	EMPTY = Low; $\overline{SI}$ = High	10	1		10	1		ns
T_{SER} Chip disable setup time[3]	↑$\overline{CE}$	↓$\overline{SO}$	EMPTY = Low; $\overline{SI}$ = High	10	1		10	1		ns
T_{PER} Chip enable delay time	↓$\overline{CE}$	↓$\overline{READ}$	EMPTY = Low; $\overline{SO}$ = Low; $\overline{SI}$ = High		64	95		64	105	ns
RESET TIMING										
T_{RR} $\overline{RESET}$ recovery	↑$\overline{RESET}$	↓$\overline{WRITE}$	$\overline{SI}$ = Low		57	75		57	80	ns
T_{RL} $\overline{RESET}$ pulse width (low)	↓$\overline{RESET}$	↑$\overline{RESET}$		25	8		25	8		ns
FULL/EMPTY OVERRIDE TIMMING:										
T_{FW} Override Recovery for FULL	↓FULL	↓$\overline{WRITE}$	Stack = Full; $\overline{SI}$ = Low; $\overline{SO}$ = High		70	95		70	110	ns
T_{ER} Override Recovery for EMPTY	↓EMPTY	↓$\overline{READ}$	Stack = EMPTY; $\overline{SO}$ = Low; $\overline{SI}$ = High		65	90		65	105	ns

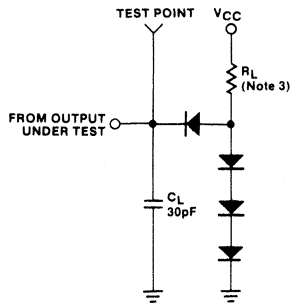
NOTES 1. Such that write/read request is inhibited after stack becomes full/empty

2. The earliest rising edge of $\overline{CE}$ such that the $\overline{WRITE}$ or $\overline{READ}$ output always occurs.

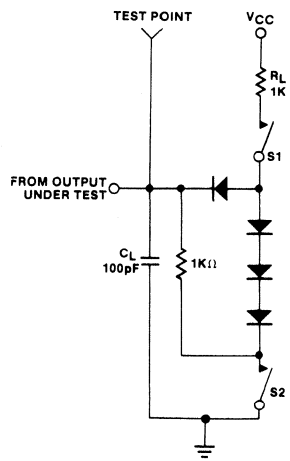
3. The latest rising edge of $\overline{CE}$ such that the $\overline{WRITE}$ or $\overline{READ}$ output never occurs.

TEST LOADING CIRCUITS

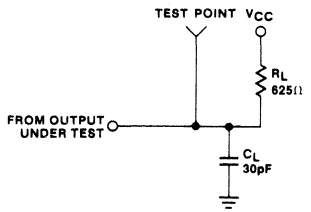
APPLICABLE PINS: $\overline{WRITE}$ (7), $\overline{READ}$ (8), HALF FULL (10)



APPLICABLE PINS: A_n (15-20, 22-27)



APPLICABLE PINS: FULL (9) AND EMPTY (11)

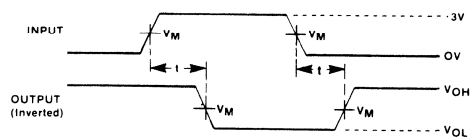


OUTPUT STATE		SWITCH POSITION	
FROM	TO	S1	S2
Low	High	Closed	Closed
High	Low	Closed	Closed
High	HI-Z	Closed	Closed
Low	HI-Z	Closed	Closed
HI-Z	High	Open	Closed
HI-Z	Low	Closed	Open

NOTES
1. In all cases C_L includes probe and jig capacitance.
2. All diodes are 1N916, 1N3064, or equivalent.
3. For $\overline{READ}$ and $\overline{WRITE}$ outputs, R_L = 280 ohms; for HALF FULL output, R_L = 2K ohms.

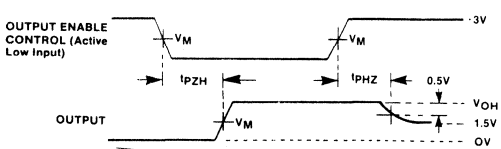
AC TEST WAVEFORMS

PROPAGATION DELAY
(Typical Example)

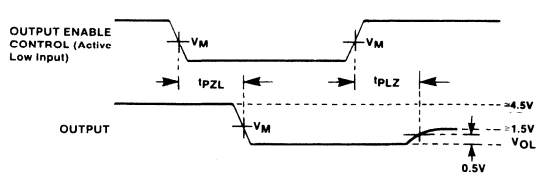


Note
Pulse widths and Setup/Hold times are measured using the same reference points as above waveform.

3-STATE ENABLE TIME TO LOW LEVEL AND DISABLE TIME FROM LOW LEVEL



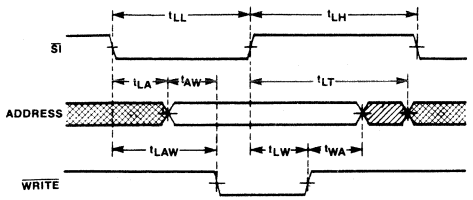
3-STATE ENABLE TIME TO HIGH LEVEL AND DISABLE TIME FROM HIGH LEVEL



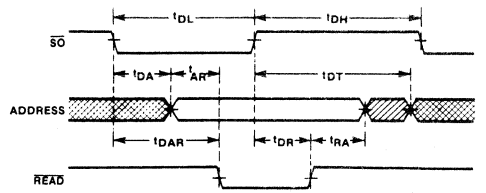
V_M = 1.5 V

TIMING DIAGRAMS

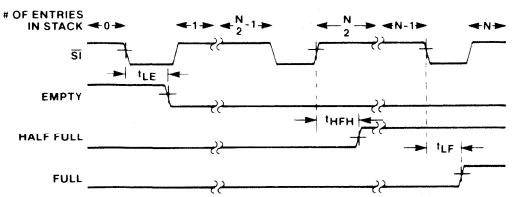
WRITE CYCLE TIMING



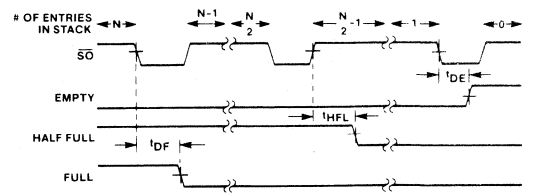
READ CYCLE TIMING



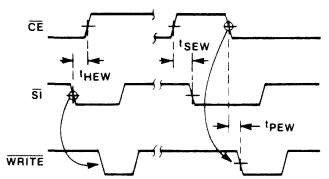
STATUS OUTPUT TIMING-WRITE



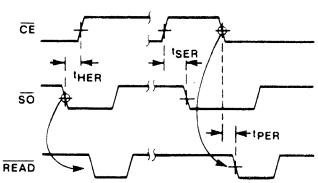
STATUS OUTPUT TIMING-READ



CHIP ENABLE TIMING WRITE*

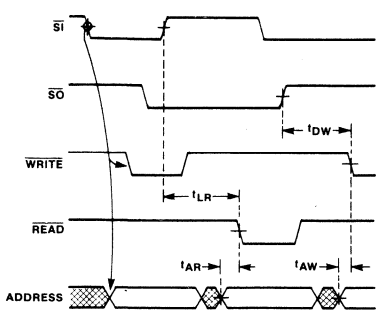


CHIP ENABLE TIMING READ*

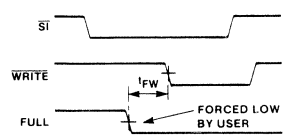


*The rising edge of $\overline{CE}$ should not occur within 10-nanoseconds before or after a falling edge of $\overline{SI}$ or $\overline{SO}$.

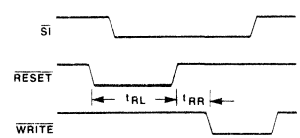
CHANGE OF CYCLE TIMING



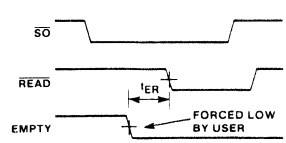
FULL OVERRIDE TIMING



RESET TIMING



EMPTY OVERRIDE TIMING



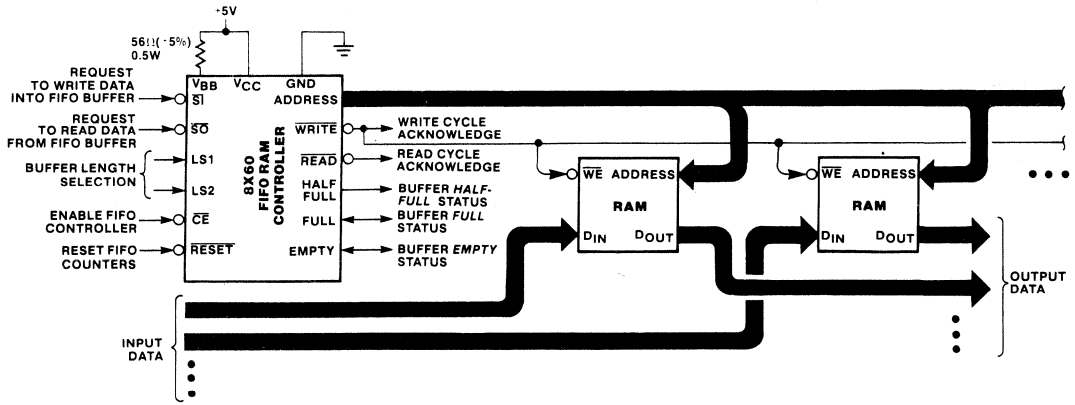
KEY

High-impedance state

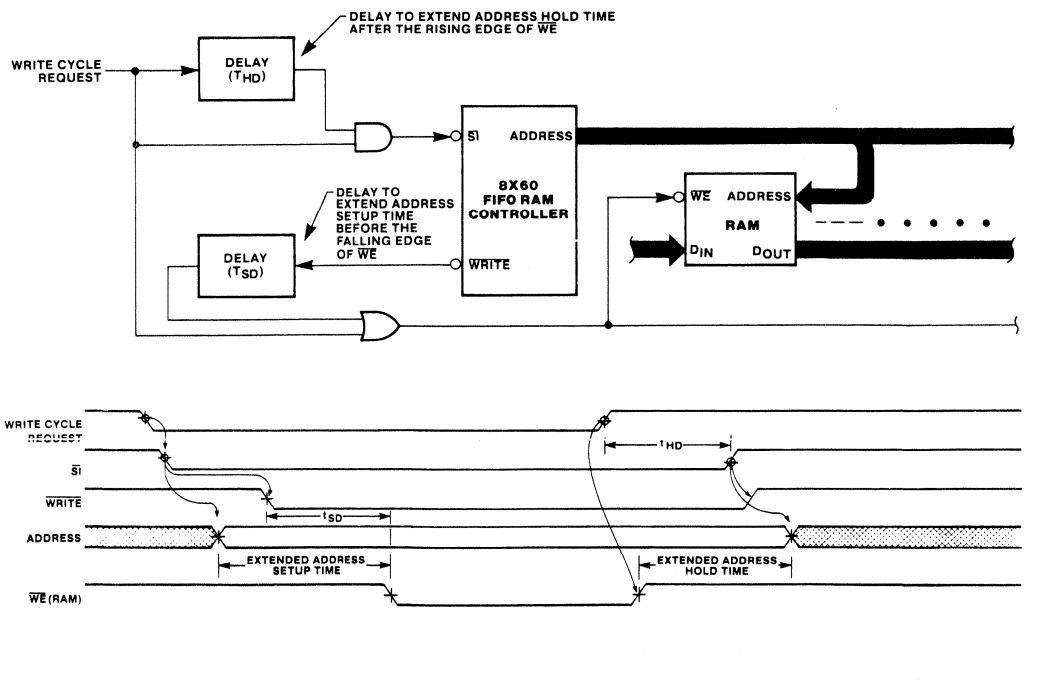
Changing data

APPLICATIONS

IMPLEMENTATION OF A FIFO BUFFER USING THE 8X60 AND HIGH SPEED RAM



USING 8X60 WITH HIGH-DENSITY MOS RAMs



Error Detection and Correction (EDC) Unit

Originally published by Signetics March 1985

PRODUCT DESCRIPTION

The Signetics 2960 Error Detection and Correction Unit (EDC) (Figure 1) contains the logic necessary to generate check bits on a 16-bit data field according to a modified Hamming Code, and to correct the data word when check bits are supplied. Operating on data read from memory, the EDC will correct any single bit error and will detect all double and some triple bit errors. For 16-bit words, 6 check bits are used. The 2960 can be expanded to operate on 32-bit words (7 check bits) and 64-bit words (8 check bits). In all configurations, the device makes the error syndrome available on separate outputs for data logging.

The Signetics 2960 also features two diagnostic modes, in which diagnostic data can be forced into portions of the chip to simplify device testing and to execute system diagnostic functions.

The product is supplied in a 48 lead hermetic DIP package and a 48-pin plastic package.

FEATURES:

- **Boosts Memory Reliability—Corrects all single-bit errors. Detects all double and some triple-bit errors. Reliability of dynamic RAM systems is increased more than 60-fold.**
- **Very High Speed—Perfect for MOS microprocessor, minicomputer and mainframe systems.**
  - **Data in to error detect: 32ns worst case.**
  - **Data in to corrected data out: 65ns worst case.**

  High performance systems can use the Signetics EDC in the check-only mode to avoid memory system slowdown.
- **Replaces 25 to 50 MSI chips—All necessary features are built-in to the Signetics 2960, including diagnostics, data in, data out and check bit latches.**
- **Handles Data Words From 8 to 64 Bits—The Signetics 2960 is cascadable: 1 EDC for 8 or 16 bits, 2 for 32 bits, 4 for 64 bits.**
- **Easy Byte Operations—Separate byte enables on the data out latch simplify the steps and cuts the time required by byte writes.**
- **Built-In Diagnostics—The processor may completely exercise the EDC under software control to check for proper operation.**

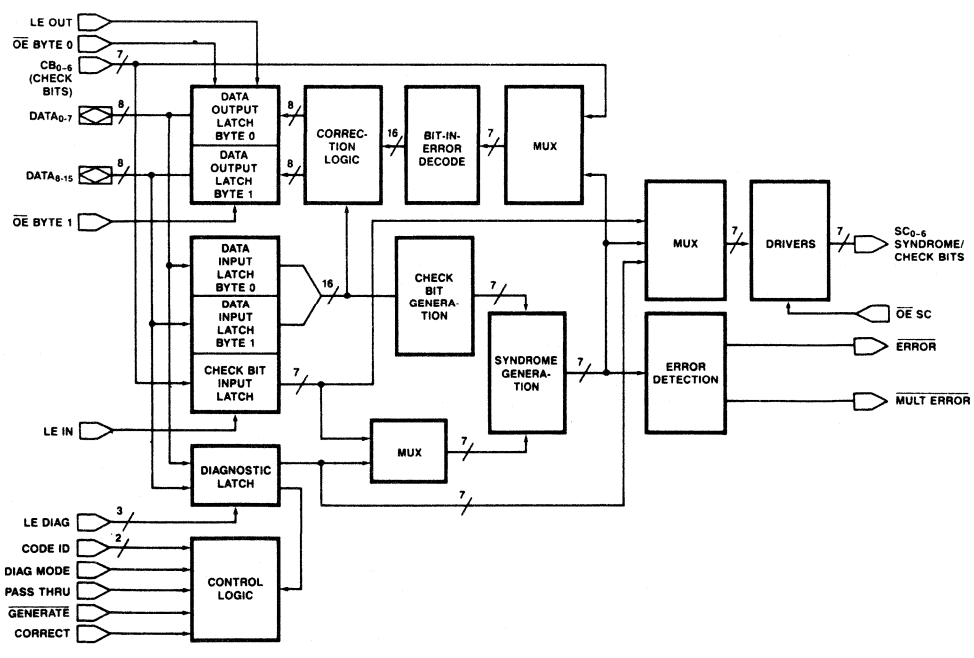
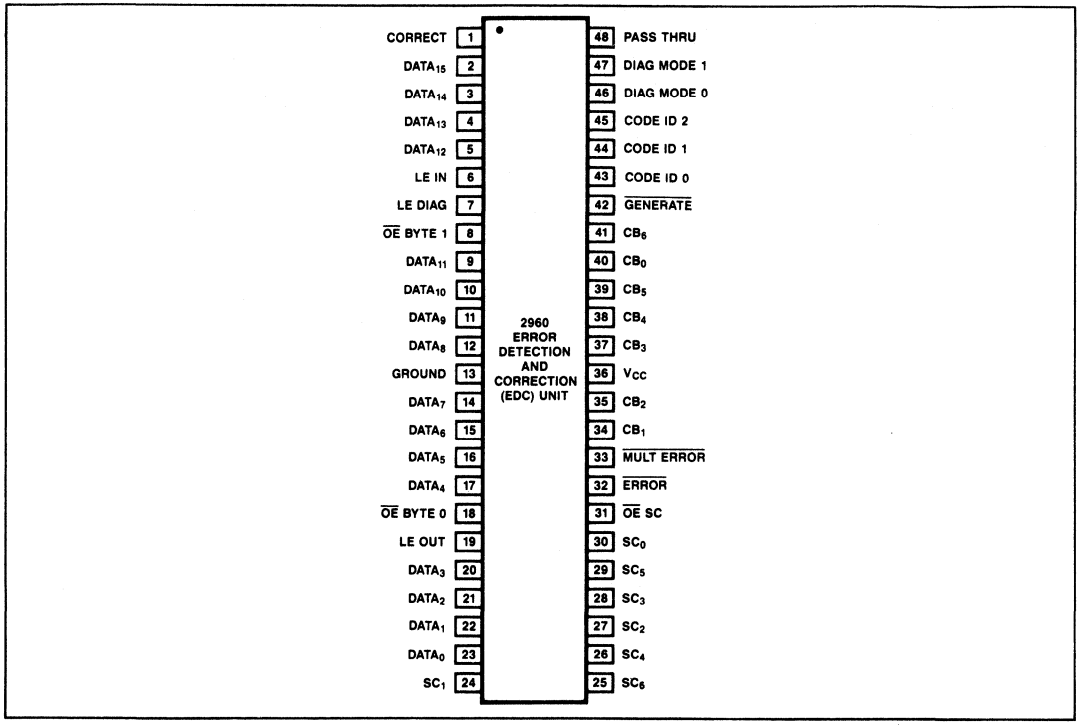


Figure 1. Block Diagram of 2960

2960 PACKAGE AND PIN DESIGNATIONS



PIN NO.	IDENTIFIER	FUNCTION
1	Correct	Correct input. When HIGH this signal allows the correction network to correct any single-bit error in the Data input Latch (by complementing the bit-in-error) before putting it onto the Data Output Latch. When LOW the EDC will drive data directly from the Data input latch to the Data Output Latch without correction.
2-5 9-12 14-17 20-23	$DATA_{15-12}$ $DATA_{11-8}$ $DATA_{7-4}$ $DATA_{3-0}$	16 bidirectional data lines. They provide input to the Data Input Latch and Diagnostic Latch, and receive output from the Data Output Latch. $DATA_0$ is the least significant bit; $DATA_{15}$ is the most significant.
6	LE IN	Latch Enable — Data Input Latch. Controls latching of the input data. When HIGH the Data input Latch and Check Bit Input Latch follow the input data and input check bits. When LOW, the Data Input Latch and Check Bit Input Latch are latched to their previous state.
7	LE DIAG	Latch Enable — Diagnostic Latch. When HIGH the Diagnostic Latch follows the 16-bit data on the input lines. When LOW the outputs of the Diagnostic Latch are latched to their previous states. The Diagnostic Latch holds diagnostic check bits, and internal control signals for CODE ID_{0-2}, DIAG $MODE_{0-1}$, CORRECT and PASS THRU.
18,8	OE BYTE 0, OE BYTE 1	Output Enable — Bytes 0 and 1, Data Output Latch. These lines control the 3-state outputs for each of the two bytes of the Data Output Latch. When LOW these lines enable the Data Output Latch and when HIGH these lines force the Data Output Latch into the high impedance state. The two enable lines can be separately activated to enable only one byte of the Data Output Latch at a time.
13	GND	Ground.
19	LE OUT	Latch Enable — Data Output Latch. Controls the latching of the Data Output Latch. When LOW the Data Output Latch is latched to its previous state. When HIGH the Data Output Latch follows the output of the Data Input Latch as modified by the correction logic network. In Correct Mode, single-bit errors are corrected by the network before loading into the Data Output Latch. In Detect Mode, the contents of the Data

PIN NO.	IDENTIFIER	FUNCTION
		Input Latch are passed through the correction network unchanged into the Data Output Latch. The inputs to the Data Output Latch are unspecified if the EDC is in Generate Mode.
24,25-30	SC_1S_6-S_0	Syndrome/Check Bit outputs. These seven lines hold the check/partial-check bits when the EDC is in Generate Mode, and will hold the syndrome/partial syndrome bits when the device is in Detect or Correct Modes. These are 3-state outputs.
31	OE SC	Output Enable — Syndrome/Check Bits. When LOW, the 3-state output lines SC_{0-6} are enabled. When HIGH, the SC outputs are in the high impedance state.
32	ERROR	Error Detected output. When the EDC is in Detect or Correct Mode, this output will go LOW if one or more syndrome bits are asserted, meaning there are one or more bit errors in the data or check bits. If no syndrome bits are asserted, there are no errors detected and the output will be HIGH. In Generate Mode, ERROR is forced HIGH. (In a 64-bit configuration, ERROR must be externally implemented.)
33	MULT ERROR	Multiple Errors Detected output. When the EDC is in Detect or Correct Mode, this output if LOW indicates that there are two or more bit errors that have been detected. If HIGH this indicates that either one or no errors have been detected. In Generate mode, MULT ERROR is forced HIGH (in a 64-bit configuration, MULT ERROR must be externally implemented.)
40, 34-35 37-39, 41	CB_{0-6}	Seven Check Bit input lines. The check bit lines are used to imput check bits for error detection. Also used to input syndrome bits for error correction in 32 and 64-bit configurations.
36	V_{CC}	+5V power supply.
42	GENERATE	Generate Check Bits input. When this input is LOW the EDC is in the Check Bit Generate Mode. When HIGH the EDC is in the Detect Mode or Correct Mode. In the Generate Mode the circuit generates the check bits or partial check bits specific to the data in the Data Input Latch. The generated check bits are placed on the SC outputs. In the Detect or Correct Modes the EDC detects single and multiple errors, and generates syndrome bits based upon the contents of the Data Input Latch and Check Bit input Latch. In Correct Mode, single-bit errors are also automatically corrected - corrected data is placed at the inputs of the Data Output Latch. The syndrome result is placed on the SC outputs and indicates in a coded form the number of errors and the bit-in-error.
43-45	Code ID_{0-2}	Code Identification inputs. These three bits identify the size of the total data word to be processed and which 16-bit slice of larger data words a particular EDC is processing. The three allowable data word sizes are 16, 32 and 64 bits and their respective modified Hamming codes are designated 16/22, 32/39 and 64/72. Special CODE ID input 001 (ID_2, ID_1, ID_0) is also used to instruct the EDC that the signals, CODE ID_{0-2}, DIAG MODE $_{0-1}$, CORRECT and PASS THRU are to be taken from the Diagnostic Latch, rather than from the input control lines.
46,47	DIAG $MODE_{0-1}$	Diagnostic Mode Select. These two lines control the initialization and diagnostic operation of the EDC.
48	PASS THRU	Pass Thru input. This line when HIGH forces the contents of the Check Bit Latch onto the Syndrome/Check Bit outputs (SC_{0-6}) and the unmodified contents of the Data Input Latch onto the inputs of the Data Output Latch.

ARCHITECTURAL SUMMARY

The EDC Unit is a powerful 16-bit cascadable slice used for check bit generation, error detection, error correction and diagnostics.

As shown in Figure 1, the device consists of the following:

- Data Input Latch
- Check Bit Input Latch
- Check Bit Generation Logic
- Syndrome Generation Logic
- Error Detection Logic
- Error Correction Logic
- Data Output Latch
- Diagnostic Latch
- Control Logic

Data Input Latch

16 bits of data are loaded from the bidirectional DATA lines under control of the Latch Enable input, LE IN. Depending on the control mode the input data is either used for check bit generation or error detection/correction.

Check Bit Input Latch

Seven check bits are loaded under control of LE IN. Check bits are used in the Error Detection and Error Correction modes.

Check Bit Generation Logic

This block generates the appropriate check bit for the 16 bits of data in the Data Input Latch. The check bits are generated according to a modifed Hamming code.

Syndrome Generation Logic

In both Error Detection and Error Correction modes, this logic block compares the check bits read from memory against a newly generated set of check bits produced for the data read in from memory. If both sets of check bits match, then there are no errors. If there is a mismatch, then one or more of the data or check bits is in error.

The syndrome bits are produced by an exclusive-OR of the two sets of check bits. If the two sets of check bits are identical (meaning there are no errors) the syndrome bits will

be all zeroes. If there are errors, the syndrome bits can be decoded to determine the number of errors and the bit-in-error.

Error Detection Logic

This section decodes the syndrome bits generated by the Syndrome Generation Logic. If there are no errors in either the input datas or check bits, the ERROR and MULT ERROR outputs remain HIGH. If one or more errors are detected, ERROR goes LOW. If two or more errors are detected, both ERROR and MULT ERROR go LOW.

Error Correction Logic

For single errors, the Error Correction Logic complements (corrects) the single data bit in error. This corrected data is loadable into the Data Output Latch, which can then be read onto the bidirectional data lines. If the single error is one of the check bits, the correction logic does not place corrected check bits on the syndrome/check bit outputs. If the corrected check bits are needed the EDC must be switched to Generate Mode.

Data Output Latch

The Data Output Latch is used for storing the result of an error correction operation. The latch is loaded from the correction logic under control of the Data Output Latch Enable, LE OUT. The Data Output Latch may also be loaded directly from the Data Input Latch under control of the PASS THRU control input.

The Data Output Latch is split into two 8-bit (byte) latches which may be enabled independently for reading onto the bidirectional data lines.

Diagnostic Latch

This is a 16-bit latch loadable from the bidirectional data lines under control of the Diagnostic Latch Enable, LE DIAG. The Diagnostic Latch contains check bit information in one byte and control information in the other byte. The Diagnostic Latch is used for driving the device when in Internal Control Mode, or for supplying check bits when in one of the Diagnostic Modes.

Control Logic

The control logic determines the specific operating mode of the device. Normally the control logic is driven by external control inputs. However, in Internal Control Mode, the control signals are read from the Diagnostic Latch.

FUNCTIONAL OPERATION

The EDC contains the logic necessary to generate check bits on a 16-bit data field according to a modified Hamming code (Table 1). Operating on data read from memory, the EDC will correct any single-bit error and will detect all double and some triple-bit errors. The EDC can be configured to operate on 16-bit data words (with 6 check bits), 32-bit data words (with 7 check bits) and 64-bit data words (with 8 check bits). In fact, the EDC can be configurd to work on data words from 8 to 64 bits. In all configurations, the device makes the error syndrome bits available on separate outputs for error data logging.

Code and Byte Specification

The EDC may be configured in several different ways and operates differently in each configuration. It is necessary to indicate to the device what size data word is involved and which bytes of the data word it is processing. This is done with input lines CODE ID_{0-2}, as shown in Table 1; the three modified Hamming codes are defined below:

- 16/22 — 16 data bits, 6 check bits, 22 bits in total.
- 32/39 code — 32 data bits, 7 check bits, 39 bits in total.
- 64/72 code — 64 data bits, 8 check bits, 72 bits in total.

CODE ID input 001 (ID_2, ID_1, ID_0) is a special code, described later, used to operate the device in Internal Control Mode.

Control Mode Selection

The device control lines are GENERATE, CORRECT, PASS THRU, DIAG $MODE_{0-1}$ and CODE ID_{0-2}. Table 2 indicates the operating modes selected by various combinations of the control line inputs.

Diagnostics

Table 3 shows specifically how DIAG $MODE_{0-1}$ select between normal operation, initialization and one of two diagnostic modes. The Diagnostic Modes allow the user to operate the EDC under software control in order to verify proper functioning of the device.

Check and Syndrome Bit Labelling

The check bits generated in the EDC are designed as follows:

- 16-bit configuration - CX, C0, C1, C2, C4, C8;
- 32-bit configuration - CX, C0, C1, C2, C4, C8, C16;
- 64-bit configuration - CX, C0, C1, C2, C4, C8, C16, C32.

Syndrome bits are similarly labeled SX through S32. There are only 6 syndrome bits in the 16-bit configuration, 7 for 32 bits and 8 syndrome bits in the 64-bit configuration.

Initialize Mode

The inputs of the Data Output Latch are forced to zeroes. The check bit outputs (SC) are generated to correspond to the all-zero data. ERROR and MULT ERROR are forced HIGH in the Initialize Mode.

Initialize Mode is useful after power up when RAM contents are random. The EDC may be placed in initialize mode and its outputs written in to all memory locations by the processor.

Code Selections

The Signetics 2960 EDC uses a modified Hamming Code which provides the following functions:

- Cascading of EDC Units
- Detection of all double-bit errors
- Detection of gross error conditions (all "0s" or all "1s").

The error correction code can be selected independent of the processor with the exception of diagnostic software.

Diagnostic software run by a processor to checkout the EDC system must know specifically which code is being used. This is only a problem when the EDC replaces an existing MSI implementation on an existing computer. In this case, the computer's software must first determine which of two codes (the old one used by the MSI implementation or the new one used by the EDC) is used by the computer's memory system.

This is easily determined by writing a test data word into memory and then examining whether the generated check bits are typical of the old or the new code. From then on the software runs only the diagnostic approporiate for the code used on that particular computer's memory system.

16-Bit Data Word Configuration

The 16-bit format consists of 16 data bits, six check bits and, as previously indicated, is designated as the 16/22 code. The data format and I/O configuration for a 16-bit word is shown in Figure 2.

Generate Mode

In this mode check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated are placed on the outputs SC_{0-5} (SC_6 is a logical one, or high).

Check bits are generated according to a modified Hamming code. Details of the code for check bit generation are contained in Table 4.

Each check bit is generated as either an XOR or XNOR of eight of the 16 data bits as indicated in the table. The XOR function results in an even parity check bit; the XNOR is an

odd parity check bit. Data flow for the Generate Mode is shown in Figure 3.

Detect Mode

In this mode the device examines the contents of the Data Input Latch against the Check Bit Input Latch, and will detect all single-bit errors, all double-bit errors and some triple-bit errors. If one or more errors are detected, ERROR goes LOW. If two or more errors are detected, MULT ERROR goes LOW. Both error indicators are HIGH if there are no errors.

Figure 2. 16-Bit Data Format and I/O Configuration

Also available on device outputs SC_{0-5} are the syndrome bits generated by the error detection step. The syndrome bits may be decoded to determine if a bit error was detected and, for single-bit errors, which of the data or check bits is in error. Table 5 provides decoding data for the syndrome bits generated by the 16-bit configuration (as an example, if the syndrome bits SX/S0/S1/S2/S4/S8 were 101001 this would be decoded to indicate that there is a single-bit error at data bit 9). If no error is detected the syndrome bits will all be zeroes.

In Detect Mode, the contents of the Data Input Latch are driven directly to the inputs of the Data Output Latch without correction.

Correct Mode

In this mode, the EDC functions the same as in Detect Mode except that the correction network is allowed to correct (complement) any single-bit error of the Data Input Latch before putting it onto the inputs of the Data Output Latch - see Figure 4. If multiple errors are detected, the output of the correction network is unspecified. If the single-bit error is a check bit there is no automatic correction. If check bit correction is desired, this can be done by placing the device in Generate Mode to produce a correct check bit sequence for the data in the Data Input Latch.

Pass Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch and the contents of the Check Bit Input Latch are placed on outputs SC_{0-5}. ERROR and MULT ERROR are forced HIGH in this mode.

Diagnostic Latch

The diagnostic Latch serves both for diagnostic uses and internal control uses. It is loaded from the DATA lines under the control of LE DIAG. Table 6 shows the loading definitions for the DATA lines.

Diagnostic Generate/Detect/Correct

These are special diagnostic modes selected by DIAG $MODE_{0-1}$ where either normal check bit inputs or outputs are substituted for by check bits loaded into the Diagnostic Latch—See Table 2 for details. Figures 5 and 6 illustrate the flow of data during the two diagnostic modes.

Internal Control Mode

This mode is selected by CODE ID_{0-2} input 001 (ID_2, ID_1, ID_0).

When in Internal Control Mode, the EDC takes the CODE ID_{0-2}, DIAG $MODE_{0-1}$, CORRECT, and PASS THRU control signals from the internal Diagnostic Latch rather than from the external input lines.

Table 6 gives the format for loading the Diagnostic Latch.

32-Bit Data Word Configuration

The 32-bit format consists of 32 data bits, seven check bits and, as previously indicated, is designated as the 32/39 code. The data format and I/O configuration for a 32-bit word is shown in Figure 7.

The upper EDC (Slice 0/1) handles the least significant bytes 0 and 1 - the external DATA lines 0 to 15 are connected to the same numbered inputs of the upper device. The lower EDC (Slice 2/3) handles the most significant bytes 2 and 3 - the external DATA lines for bits 16 to 31 are connected to inputs $DATA_0$ through $DATA_{15}$ respectively.

The valid syndrome and check bit outputs are those of Slice 2/3 as shown in the diagram. In Correct Mode these must be read into Slice 0/1 via the CB inputs and are selected by the MUX as inputs to the bit-in-error decoded (see block diagram), thus requiring external buffering and output enabling of the check bit lines as shown. The OE SC signal can be used to control enabling of check bit inputs - when syndrome outputs are enabled, the external check bit inputs will be disabled.

The valid ERROR and MULT ERROR outputs are those of the Slice 2/3. The ERROR and MULT ERROR outputs of Slice 0/1 are unspecified. All of the latch enables and control signals must be input to both of the devices.

Generate Mode

In this mode check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated are placed on the outputs SC_{0-6} of Slice 2/3.

Check bits are generated according to a modified Hamming code. Details of the code for check bit generation are contained in Table 7. Check bits are generated as either an XOR or XNOR or 16 of the 32 data bits as indicated in the table. The XOR function results in an even parity check bit, the XNOR in an odd parity check bit.

Detect Code

In this mode the device examines the contents of the Data Input Latch against the Check Bit Input Latch, and will detect all single-bit errors, all double-bit errors and some triple-bit errors. If one or more errors are detected, ERROR goes LOW. If two or more errors are detected MULT ERROR goes LOW. Both error indicators are HIGH if there are no errors. The valid ERROR and MULT ERROR signals are those of Slice 2/3 - those of Slice 0/1 are undefined.

Also available on Slice 2/3 outputs $SC_{0/6}$ are the syndrome bits generated by the error detection step. The syndrome bits may be decoded to determine if a bit error was detected and, for single-bit errors, which of the data or check bits is in error. Table 8 gives the chart for decoding the syndrome bits generated for the 32-bit configuration (as an example, if the syndrome bits SX/S0/S1/S2/S4/S8/S16 were 0010011 this would be decoded to indicate that there is a single-bit error at data bit 25). If no error is detected the syndrome bits will be all zeroes.

In Detect Mode, the contents of the Data Input Latch are driven directly to the inputs of the Data Output Latch without corrections.

Correct Mode

In this mode, the EDC functions the same as in Detect Mode except that the correction network is allowed to correct (complement) any single-bit error of the Data Input Latch before putting it onto the inputs of the Data Output Latch. If multiple errors are detected, the output of the correction network is unspecified. If the single-bit error is a check bit there is no automatic correction - if desired this would be done by placing the device in Generate Mode to produce a correct check bit sequence for the data in the Data Input Latch.

For data correction, both Slices 0/1 and 2/3 require access to the syndrome bits on Slice 2/3's outputs SC_{0-6}. Slice 2/3 has access to these syndrome bits through internal data paths, but for Slice 0/1 they must be read through the inputs CB_{0-6}. The device connections for this are shown in Figure 7. When in Correct Mode the SC outputs must be enabled so that they are available for reading in through the CB inputs.

Pass Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch and the contents of the

Check Bit Input Latch are placed on outputs SC_{0-6} of Slice 2/3. ERROR and MULT ERROR are forced HIGH in this mode.

Diagnostic Latches/AC Calculations

Table 9 shows how the latches (Slice 1 and Slice 2) are loaded for code 32/39 (32-bit format). Table 10 shows key AC parameters for the 32-bit configuration.

64-Bit Data Word Configuration

The 64-bit format consists of 64 data bits, eight check bits and, as previously indicated, is designated as the 64/72 code. The data format and I/O configuration for a 64-bit word is shown in Figure 8.

The configuration to process the 64-bit format is similar to that shown in Figure 2. In this configuration a portion of the syndrome generation and error detection is implemented externally of the EDCs in MSI. For error correction the syndrome bits generated must be read back into all four EDCs through the CB inputs. This necessitates the check bit buffering shown in Figure 8. The OE SC signal can control the check bit enabling - when syndrome bit outputs are enabled the external check bit lines will be disabled so that the syndrome bits may be read onto the CB inputs.

The error detection signals for the 64-bit configuration differ from the 16 and 32-bit configurations. The ERROR signal functions the same: it is LOW if one or more errors are detected, and HIGH if no errors are detected. The DOUBLE ERROR signal is HIGH if and only if a double-bit error is detected - it is LOW otherwise. All of the MULT ERROR outputs of the four devices are valid. MULT ERROR is LOW for all three ERROR cases and some DOUBLE ERROR combinations - See Table 15. It is HIGH if either zero or one errors are detected.

This is a different meaning for MULT ERROR than in other configurations.

Generate Mode

In this mode check bits will be generated that correspond to the contents of the Data Input Latch. The check bits generated appear at the outputs of the XOR gates as indicated in Figure 8.

Check bits are generated according to a modified Hamming code. Details of the code for check bit generation are contained in Table 11. Check bits are generated as either an XOR or XNOR of 32 of the 64 bits as indicated in the table. The XOR function results in an even parity check bit, the XNOR in an odd parity check bit.

Detect Mode

In this mode the device compares the contents of the Data Input Latch against the contents of the Check Bit Input Latch and will detect all single-bit errors, all double-bit errors and some triple-bit errors. If one or more errors are detected, ERROR goes LOW. If exactly two errors are detected, DOUBLE ERROR goes HIGH. If three or more errors are detected, MULT ERROR goes LOW - the MULT ERROR output of any of the four EDCs may be used.

Available as XOR gate outputs are the generated syndrome bits - see Figure 8. The syndrome bits may be decoded to determine if a bit error was detected and for single-bit errors, which of the data or check bits is in error. Table 12 gives the chart for encoding the syndrome bits generated for the 64-bit configuration (as an example, if the syndrome bits SX/S1/S2/S4/S8/S16/S32 were 00100101 this would be decoded to indicate that there is a single-bit error at data bit 41). If no error is detected the syndrome bits will all be zeroes.

In Detect Mode the contents of the Data Input Latch are driven directly to the Inputs of the Data Output Latch without corrections.

Correct Mode

In this mode, the EDC functions the same as in Detect Mode except that the correction network is allowed to correct (complement) any single-bit error of the Data Input Latch before putting it onto the inputs of the Data Output Latch. If multiple errors are detected, the output of the correction network is unspecified, if the single bit error is a check bit there is no automatic correction. Check bit correction can be done by placing the device in generate mode to produce a correct check bit sequence for the data in the Data Input Latch.

To perform the correction step, all four slices require access to the syndrome bits which are generated externally of the devices. This access is provided by reading the syndrome bits in through the CB inputs where they are selected as inputs to the bit-in-error decoded by the multiplexer (see block diagram). The device connections for this operation are shown in Figure 8. When in Coorect Mode the SC outputs must be enabled so that the syndrome bits are available at the CB inputs.

Pass Thru Mode

In this mode, the unmodified contents of the Data Input Latch are placed on the inputs of the Data Output Latch, and the contents of the Check Bit Input Latch are passed through the external XOR network and appear inverted at the XOR gate ouputs labeled CX to C32 - see Figure 8.

Diagnostic Latch

The Diagnostic Latch is used for both diagnostic and internal control of the EDC. Table 13 provides bit definitions and shows the 64-bit loading format.

Diagnostic Generate/Detect/Correct

These are special diagnostic modes selected by DIAG $MODE_{0-1}$ where either normal check bit inputs or outputs are substituted for by check bits from the Diagnostic Latch - see Table 2 for details.

Internal Control Mode

This mode is selected by CODE ID_{0-2}, input 001 (ID_2, ID_2, ID_0).

When in Internal Control Mode the EDC takes the CODE ID_{0-2}, DIAG MODE $_{0-1}$, CORRECT and PASS THRU signals from the internal Diagnostic Latch rather than from the external control lines - see Table 13 for latch loading.

AC Calculations

Table 14 shows key AC parameters for the 64-bit configuration.

Functional Equations

The following equations and tables describe in detail how the output values of the Signetics 2960 are determined as a function of input values and internal states of the chip. Before examining the tables, the following symbol definitions should be carefully studied.

Definitions

D_i – ($DATA_i$ if LE IN is HIGH or the output of bit i of the Data input Latch if LE IN is LOW)
C_i – (CB_i if LE IN is HIGH or the output of bit i of the Check Bit Latch if LE IN is LOW)
DL_i – Output of bit i of the Diagnostic Latch
S_i – Internally generated syndromes (same as outputs of SC_i if outputs enabled)
PA – D0 ⊕ D1 ⊕ D2 ⊕ D4 ⊕ D6 ⊕ D8 ⊕ D10 ⊕ D12
PB – D0 ⊕ D1 ⊕ D2 ⊕ D3 ⊕ D4 ⊕ D5 ⊕ D6 ⊕ D7
PC – D8 ⊕ D9 ⊕ D10 ⊕ D11 ⊕ D12 ⊕ D13 ⊕ D14 ⊕ D15
PD – D0 ⊕ D3 ⊕ D4 ⊕ D7 ⊕ D9 ⊕ D10 ⊕ D13 ⊕ D15
PE – D0 ⊕ D1 ⊕ D5 ⊕ D6 ⊕ D7 ⊕ D11 ⊕ D12 ⊕ D13
PF – D2 ⊕ D3 ⊕ D4 ⊕ D5 ⊕ D6 ⊕ D7 ⊕ D14 ⊕ D15
PG_1 – D0 ⊕ D4 ⊕ D6 ⊕ D7
PG_2 – D1 ⊕ D2 ⊕ D3 ⊕ D5
PG_3 – D8 ⊕ D9 ⊕ D11 ⊕ D14
PG_4 – D10 ⊕ D12 ⊕ D13 ⊕ D15

Error Signals

$$\text{ERROR} \leftarrow (\overline{S6 \bullet (ID_1 + ID_2)}) \bullet \overline{S5} \bullet \overline{S4} \bullet \overline{S3} \bullet \overline{S2} \bullet \overline{S1} \bullet \overline{S0} + \text{GENERATE} + \text{INITIALIZE} + \text{PASSTHRU}$$

$$\overline{\text{MULT ERROR}}\ (16 \text{ and } 32\text{-Bit Modes}) \leftarrow \overline{((\overline{S6 \bullet ID_1}) \bullet S5 \bullet S4 \bullet S3 \bullet S2 \bullet S1 \bullet S0)(\overline{\text{ERROR}})} + \text{TOME} + \text{GENERATE} + \text{PASSTHRU} + \text{INITIALIZE}$$

$$\overline{\text{MULT ERROR}}\ (64\text{-Bit Modes}) \leftarrow \text{TOME} + \text{GENERATE} + \text{PASSTHRU} + \text{INITALIZE}$$

SC Outputs

Tables 16 through 20 show how outputs $SC_{0\text{-}6}$ are generated in each control mode for various CODE IDs (internal control mode not applicable).

Data Correction

Tables 21 through 27 show which data output bits are corrected (inverted) depending upon the syndromes and the CODE ID position. Note that the syndromes that determine data correction are in some cases syndromes input externally via the CB inputs and in some cases syndromes generated internally by that EDC (S_i are the internal syndromes and are the same as the value of the SC_i output of that EDC if enabled).

The tables show the number of data bit inverted (corrected) if any for the CODE ID and syndrome combination.

SYSTEM DESIGN CONSIDERATIONS

High Performance Parallel Operation

For maximum memory system performance the EDC should be used in the Check-Only configuration shown in Figure 9. With this configuration the memory system operates as fast with EDC as it would without.

On reads from memory, data is read out from the RAMs directly to the data bus (same as in a non-EDC system). At the same time, the data is read into the EDC to check for errors. If an error exists the EDC's error flags are used to interrupt the CPU and/or to stretch the memory cycle. If no error is detected, no slowdown is required.

If an error is detected, the EDC generates corrected data for the processor. At the designer's option the correct data may be written back into memory; error logging and diagnostic routines may also be run under processor control.

The Check-Only configuration allows data reads to proceed as fast with EDC as without. Only if an error is detected is there any slowdown. But even if the memory system had an error every hour this would mean only one error every 3-4 billion memory cycles. So even with a very high error rate, EDC in a Check-Only configuratin has essentially zero impact on memory system speed.

On writes to memory, check bits must be generated before the full memory word can be written into memory. The data word is frequently buffered while the check bits are generated. This makes the check bit generate time transparent to the processor.

EDC in the Data Path

The simplest configuration for EDC is to have the EDC directly in the data path as shown in Figure 10. Correct-Always Configuration). In the configuration data read from memory is always corrected prior to putting the data on the data bus. The advantages are simpler operation and no need for mid-cycle interrupts. The disadvantages is that memory system speed is slowed by the amount of time it takes for error correction on every cycle.

Usually the Correct-Always Configuration will be used with MOS microprocessors which have ample memory timing budgets. Most high performance processors will use the high performance parallel configuration shown in Figure 9.

Scrubbing Avoids Double Errors

Single-bit errors are by far the most common in a memory system and are always correctable by the EDC.

Double bit memory errors are far less frequent than single bit errors (50 to 1, or 100 to 1) and are always detected by the EDC but not corrected.

In a memory system, soft errors occur only one at a time. A double bit error in a data word occurs when a single soft error is left uncorrected and is followed by another error in the data word hours, days, or weeks after the first.

"Scrubbing" memory periodically avoids almost all double-bit errors. In the scrubbing operation, every data word in a memory is periodically checked by the EDC for single-bit errors. If one is found, it is corrected and the data word written back into memory. Errors are not allowed to pile up and so most double-bit errors are avoided.

The scrubbing operation is generally done as a background routine when the memory is not being used by the processor.

If memory is scrubbed frequently, errors are detected and corrected during processor accesses need not be immediately written back into memory. Instead the error will be

corrected in memory during scrubbing. This reduces the time delay involved in a processor access of an incorrect memory word.

Correction of Double-Bit Errors

In some cases, double-bit memory errors can be corrected. This is possible when one of the two bit errors is a hard error.

When a double bit error is detected the data word should be checked to determine if one of the errors is a hard error. If so the hard error bit may be corrected by inverting it leaving only a single, correctable error. The time for this operation is negligible since it will occur infrequently.

The procedure after detection of a double error is as follows:

- Invert the data bits read from memory.
- Write the inverted data back into the same memory word.
- Re-read the memory location and XOR the newly read out value with the old. If there is no hard error then the XOR result will be 1's. If there is a hard error, it will have the same bit value regardless of what was written in. So it will show as a 0 after the XOR operation.
- Invert the hard error bit (this will "correct" it) leaving only one error in the data.
- The EDC can then correct the single bit error.
- Rewrite the correct data word into memory. This does not change the hard error but does eliminate the soft error. So the next memory access will find only a single-bit, correctable error.

An example helps to illustrate the procedure

Example of Double Bit Error Correction When One is a Hard Error

		16 DATA BITS	6 CHECK BITS
1)	Data read from memory (D_2)	1111111100000011	011010
2)	EDC detects a multiple error Syndromes:		011000
3)	Syndrome decode indicates a double bit error.		
4)	Invert the bits read from memory (D_1)	0000000011111100	100101
5)	Write D_1 back to the same memory location.		
6)	Read back the memory location (D_2)	0000000011111101	100101
7)	XOR D_1 and D_2.	1111111100000010	111111
8)	So the last data bit is the hard error. Use this to modify D_1	1111111100000010	011010
9)	Pass the modified D_1 through the EDC. The EDC detects a single bit correctable error and outputs corrected data.	1111111100000000	011010
10)	Write the corrected data back to memory to fix the soft error		

Error Logging and Preventative Maintenance

The effectiveness of preventative maintenance can be increased by logging information on errors detected by the EDC. This is called error logging.

The EDC provides syndromes when errors are detected. The syndromes indicate which bit is in error. In most memory systems, each individual RAM supplies only one bit of the memory word. So the syndrome and data word address specify which RAM was in error.

Typically a permanent/hard RAM failure is preceded by a period of time where the RAM displays an increasing frequency of intermittent, soft errors. Error logging statistics can be used to detect an increasing intermittent error frequency so that the RAM can be replaced before a permanent failure occurs.

Error logging also records the location of already hard failed RAMs. With EDC a hard failure will not halt system operation. EDC always can correct single bit errors even if it is a hard error. EDC can also correct double bit errors where one is hard and one soft. The ability to continue operation despite hard errors can greatly reduce the need for emergency field maintenance. The hard-failed RAMs can be instead replaced at low cost during a regularly scheduled preventative maintenance session.

Reducing Check Bit Overhead

Memory word widths need not be same as the data word width of the processor. There is a substantial reduction in check bit overhead if wider memory words are used.

MEMORY WORD		
#DATA BITS	#CHECK BITS	CHECK BIT OVERHEAD
8	5	38%
16	6	27%
32	7	14%
64	8	11%

This reduction in check-bit overhead lowers cost and increases the amount of data that can be packed on to each board.

The tradeoff is that when writing data pieces into memory that are narrower than the memory word width, more steps are required. These steps are exactly the same as those described in Byte Write in the Applications section. No penalty exists for reads from memory.

EDC Per Board vs EDC Per System

The choice of an EDC per system or per board depends on the economics and the architecture of the system.

Certainly the cheaper approach is to have only one EDC per system and this is a viable solution if only one memory location is accessed at a time.

This solution does require that the system have both data and check bit lines - see Figure 11. This makes retrofitting a system difficult and creates complications if static or ROM memory, which do not require check bits, are mixed in with dynamic RAM.

If the system has an advanced architecture it is quite likely that it is necessary to simultaneously access memory locations on different memory boards - see Figure 12. Architectural features that require this are interleaved memory, cache memory, and DMA that is done simultaneously with processor memory accesses. EDC per board is a simpler system from a design standpoint.

The EDC is designed to work efficiently in either the per system or per board configurations.

Test Information

Incoming test procedures on this device should be carefully planned, taking into account the complexity and power levels of the part. The following notes may be useful.

1. Insure the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400mA in 5-8ns. Inductance in the ground cable may allow the ground pin at the device to rise by 100's of millivolts momentarily.
4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. Signetics recommends using V_{IL} 0.4V and V_{IH} - 2.4V for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. To assist in testing, Signetics offers documentation on our test procedures and, in most cases, can provide Fairchild Sentry programs, under license.

MAXIMUM RATINGS (above which the useful life may be impaired)

PARAMETER	RATING	UNITS
Storage Temperature	−65 to +150	°C
Temperature (Case) Under Bias	−55 to +125	°C
Supply Voltage to Ground Potential	−0.5 to +7.0	V
DC Voltage Applied to Outputs for High Output State	−0.5 to V_{CC} max.	V
DC Input Voltage	−0.5 to +5.5	V
DC Output Current, Into Outputs	30	mA
DC Input Current	−30 to +5.0	mA

OPERATING RANGE

PART NO.	TEMPERATURE	V_{CC}
N2960N N2960I	T_A = 0 to +70°C	5V (±5%)

DC CHARACTERISTICS V_{CC} MIN = 4.75V, V_{CC} MAX = 5.25V

PARAMETER		TEST CONDITIONS[1]			2960 Min	2960 Typ[2]	2960 Max	UNITS
V_{OH}	Output HIGH Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	I_{OH} = −0.8mA		2.7			V
V_{OL}	Output LOW Voltage	V_{CC} = MIN, V_{IH} = V_{IH} or V_{IL}	I_{OL} = 8mA				0.5	V
V_{IH}	Input HIGH Voltage	Guaranteed Input Logical HIGH Voltage for all Inputs[6]			2.0			V
V_{IL}	Input LOW Voltage	Guaranteed Input Logical LOW Voltage for all Inputs[6]					0.8	V
V_I	Input Clamp Voltage	V_{CC} = MIN, I_{IN} = −18mA					−1.5	V
I_{IL}	Input LOW Current	V_{CC} = MAX V_{IN} = 0.5V	$DATA_{0-15}$				−410	μA
			All Other Inputs				−360	
I_{IH}	Input HIGH Current	V_{CC} = MAX V_{IN} = 2.7V	$DATA_{0-15}$				70	μA
			All Other Inputs				50	
I_I	Input HIGH Current	V_{CC} = MAX, V_{IN} = 5.5V					1.0	mA
I_{OZH}[4] I_{OZL}[4]	Off State (High Impedance) Output Current	V_{CC} = MAX	$DATA_{0-15}$	V_O = 2.4V			70	μA
				V_O = 0.5V			−410	
			SC_{0-6}	V_O = 2.4V			50	
				V_O = 0.5V			−50	
I_{OS}	Output Short Circuit Current[3]	V_{CC} = V_{CC} MAX + 0.5V, V_O = 0.5V			−25		−85	mA
I_{CC}	Power Supply Current[5]	V_{CC} = MAX	T_A = 25°C			300	360	mA
			T_A = 0 to +70°C					
			T_A = +70°C					

NOTES:
1. For conditions shown as MIN or MAX, use the appropriate value specified under Electrical Characteristics for the applicable device type.
2. Typical limits are at V_{CC} = 5.0V, 25°C ambient and maximum loading.
3. Not more than one output should be shorted at a time. Duration of the short circuit test should not exceed one second.
4. These are three-state outputs internally connected to TTL inputs. Input Characteristics are measured with output enables HIGH.
5. Worst case I_{CC} is at minimum temperature.
6. These input levels provide zero noise immunity and should only be tested in a static, noise-free environment.

GUARANTEED PERFORMANCE OVER COMMERCIAL TEMPERATURE RANGE OF 0 TO +70°C

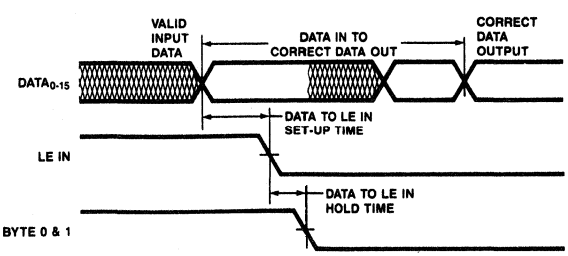
The tables that follow specify the guaranteed performance of the 2960 over the commercial operating range of 0 to +70°C, with V_{CC} from 4.75V to 5.25V. All data are in ns with inputs switching between 0V and 3V at 1V/ns and measurements made at 1.5V. All outputs have maximum DC load.

This data applies to the following part numbers: N2960N and N2960I.

COMBINATIONAL PROPAGATION DELAYS — C_L = 50pF

FROM INPUT \ TO OUTPUT	SC_{0-6}	$DATA_{0-15}$	$\overline{ERROR}$	$\overline{MULT\ ERROR}$
$DATA_{0-15}$	32	65*	32	50
CB_{0-6} (CODE ID_{2-0} 000, 011)	28	56	29	47
CB_{0-6} (CODE ID_{2-0} 010, 100, 101, 110, 111)	28	45	29	34
$\overline{GENERATE}$	35	63	36	55
CORRECT (Not Internal Control Mode)	—	45	—	—
DIAG MODE (Not Internal Control Mode)	50	78	59	75
PASS THRU (Not Internal Control Mode)	36**	44	29	46
CODE ID_{2-0}	61	90	60	80
LE IN (From latched to transparent)	39	72*	39	59
LE OUT (From latched to transparent)	—	31	—	—
LE DIAG (From latched to transparent; Not Internal Control Mode)	45	78	45	65
Internal Control Mode: LE DIAG (From latched to transparent)	67	96	66	86
Internal Control Mode: $DATA_{0-15}$ (Via Diagnostic Latch)	67	96	66	86

*Data In (or LE In) to Correct Data Out measurement requires timing as shown below.



**Device not tested in GENERATE mode.

SET-UP AND HOLD TIMES RELATIVE TO LATCH ENABLES

FROM INPUT	TO (LATCHING UP DATA)	SET-UP TIME	HOLD TIME
$DATA_{0-15}$	LE IN	6	7
CB_{0-6}	LE IN	5	6
$DATA_{0-15}$	LE OUT	44	5
CB_{0-6} (CODE ID 000, 011)	LE OUT	35	0
CB_{0-6} (CODE ID 010, 100, 101, 110, 111)	LE OUT	27	0
$\overline{GENERATE}$	LE OUT	42	0
CORRECT	LE OUT	26	1
DIAG MODE	LE OUT	69	0
PASS THRU	LE OUT	26	0
CODE ID_{2-0}	LE OUT	81	0
LE IN	LE OUT	51	5
$DATA_{0-15}$	LE DIAG	6	8

OUTPUT ENABLE/DISABLE TIMES

Output disable tests performed with C_L = 5pF and measured to 0.5V change of output voltage level.

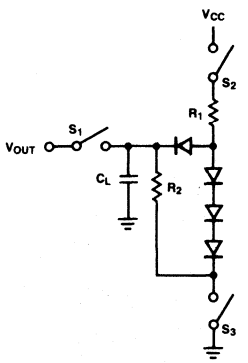
INPUT	OUTPUT	ENABLE	DISABLE
$\overline{OE}$ BYTE 0, $\overline{OE}$ BYTE 1	$DATA_{0-15}$	30	30
$\overline{OE}$ SC	SC_{0-6}	30	30

MINIMUM PULSE WIDTHS

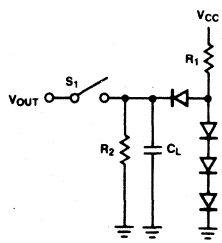
LE IN, LE OUT, LE DIAG	15

TEST LOADING

Three-State Outputs:



Normal Outputs:



TEST OUTPUT LOADS

PIN #	PIN LABEL	TEST CIRCUIT	R_1	R_2
—	D_0–D_{15}	Fig. 11	430Ω	1kΩ
24–30	SC_0–SC_6	Fig. 11	430Ω	1kΩ
32	$\overline{ERROR}$	Fig. 12	470Ω	3kΩ
33	$\overline{MULTERROR}$	Fig. 12	470Ω	3kΩ

NOTES:
1. C_L = 50pF includes scope probe, wiring and stray capacitances without device in test fixture.
2. S_1, S_2, S_3 are closed during function test and all AC tests, except output enable tests.
3. S_1 and S_3 are closed while S_2 is open for t_{PZH} test.
   S_1 and S_2 are closed while S_3 is open for t_{PZL} test.
4. R_2 = 1K for three-state output.
   R_2 is determined by the I_{OH} at V_{OH} = 2.4V for non-three-state outputs.
5. R_1 is determined by I_{OL} (MIL) with V_{CC} = 5.0V minus the current to ground through R_2.
6. C_L = 5.0pF for output disable tests.

PACKAGE DATA

Type: Plastic and Ceramic
Configuration: DIP
Width: C = 0.6" P = 0.55"
Length: 2.4"
Pin Centers: 2.54 BSC

ORDERING INFORMATION

Commercial:
N2960N (Plastic)
N2960I (Ceramic)

APPLICATIONS

Byte Write

Byte operations are increasingly common for 16 and 32-bit processors. These complicate memory operations because check bits are generated for a complete 16 or 32 or 64-bit memory word, not for a single byte.

To write a byte into memory with EDC requires the following steps — See Figures 13 and 14.

- Latch the byte into the bus buffers
- Read the complete data word from memory
- Correct the complete data word if necessary
- Insert the byte to be written into the data word
- Generate new check bits for the entire data word
- Store the data word back into memory.

(In fact these steps must be taken for any piece of data being written into memory that is not as wide as a full memory word).

The EDC is designed with the intent of keeping byte operations simple in error detection/correction systems. The EDC has separate output enables for each byte in the Data Output Latch. As shown in Figures 13 and 14, this allows the data word to be read from memory, the new byte to be inserted among the old, and new check bits to be generated using less time and less hardware than if separate byte enables were not available.

Diagnostics

EDC is used to boost the reliability of the overall system. It is necessary to also be able to check the operation of the EDC itself. For this reason the EDC has an internal control mode, a diagnostic latch, and two diagnostic modes.

To check that the EDC is functioning properly, the processor can put the EDC under software control by setting CODE ID_{2-0} to 001. This puts the EDC into Internal Control Mode. In Internal Control Mode the EDC is controlled by the contents of the Diagnostic Latch which is loaded from the DATA inputs under processor control.

The EDC is set into CORRECT Mode. The processor loads in a known set of check bits into the Diagnostic Latch, a known set of data bits into the Data In Latch, and forces data errors. The output of the EDC (syndromes, error flags, corrected data) is then compared against the expected responses. By exercising the EDC with a string of data/check combinations and comparing the output against the expected responses, the EDC can be fully checked out.

Eight Bit Data Word

Eight bit MOS microprocessors can use EDC too. Only five check bits are required. The EDC configuration for eight bits is shown in Figure 15. It operates as does the normal 16-bit configuration with the upper byte fixed at 0.

Other Word Widths

EDC on data words other than 8, 16, 32, of 64 bits can be accomplished with the 2960. In most cases the extra data bits can be forced to a constant and EDC will proceed as normal. For example a 24-bit data word is shown in Figure 16.

Single Error Correction Only

The EDC normally corrects all single bit errors and detects all double bit and some triple bit errors. To save one check bit per word the ability to detect double bit errors can be sacrificed - single errors are still detected and corrected.

Figure 17 shows single error correction only configurations for 8, 16, 32, and 64-bit data words respectively.

Check Bit Correction

The EDC detects single bit errors whether the error is a data bit or a check bit. Data bit errors are automatically corrected by the EDC. To generate corrected check bits once a single check bit error is detected, the EDC need only be switched to GENERATE mode (data in the DATA INPUT LATCH is valid).

The syndromes generated by the EDC may be decoded to determine whether the single bit error is a check bit.

In many memory systems, a check bit error will be ignored on the memory read and corrected during a periodic "scrubbing" of memory - see System Design Considerations).

Mulitiple Errors

The bit-in-error decode logic uses syndrome bits SO through S32 to correct errors, SX is only used in developing the multiple error signal. This means that some multiple errors will cause a data bit to be inverted.

For example, in the 16-bit mode if data bits 8 and 13 are in error the syndrome 111100 (SX, S0, S1, S2, S4 S8) is produced. This is flagged a double error by the error detection logic, but the decoded bit-in-error only receives syndrome 11100 (S0, S1. S2, S4, S8) which it decodes as a single error in data bit 0 and inverts that bit. If it is desired to inhibit this inversion. the multiple error output may be connected to the correct input as in Figure 18. This will inhibit correction when a multiple error occurs. Extra time delay may be introduced in the data to correct data path when this is done.

DATA BITS	CHECK BITS REQUIRED	
	SINGLE ERROR CORRECTION ONLY	SINGLE ERROR CORRECT & DOUBLE ERROR DETECT
8	4	5
16	5	6
32	6	7
64	7	8

Table 1. Hamming Code and Slice Identification

CODE ID_2	CODE ID_1	CODE ID_0	HAMMING CODE AND SLICE SELECTED
0	0	0	Code 16/22
0	0	1	Internal Control Mode
0	1	0	Code 32/39, Bytes 0 and 1
0	1	1	Code 32/39, Bytes 2 and 3
1	0	0	Code 64/72, Bytes 0 and 1
1	0	1	Code 64/72, Bytes 2 and 3
1	1	0	Code 64/72, Bytes 4 and 5
1	1	1	Code 64/72, Bytes 6 and 7

Table 2. EDC Operating Modes

OPERATING MODE	DIAGNOSTIC MODE**		$\overline{\text{GENERATE}}$	
	DM_1	DM_0	0	1
Normal	0	0	Generate	Correct*
Diagnostic Generate	0	1	Diagnostic Generate	Correct*
Diagnostic Correct	1	0	Generate	Diagnostic Correct*
Initialize	1	1	Initialize	Initialize
Pass Thru	When PASS THRU is asserted the Operating Mode is defaulted to the Pass Thru Mode.			

*Correct if the CORRECT Input is HIGH, Detect if the CORRECT Input is LOW.
**In Code ID_{2-0} 001 (ID_2, ID_1, ID_0) DM_1 and DM_0 are taken from the Diagnostic Latch.

Table 3. Diagnostic Mode Control

DIAG $MODE_1$	DIAG $MODE_0$	DIAGNOSTIC MODE SELECTED
0	0	**Non-diagnostic mode.** The EDC functions normally in all modes.
0	1	**Diagnostic Generate.** The contents of the Diagnostic Latch are substituted for the normally generated check bits when in the Generate Mode. The EDC functions normally in the Detect or Correct modes.
1	0	**Diagnostic Detect/Correct.** In the Detect or Correct Mode, the contents of the Diagnostic Latch are substituted for the check bits normally read from the Check Bit Input Latch. The EDC functions normally in the Generate Mode.
1	1	**Initialize.** The outputs of the Data Input Latch are forced to zeroes (and latched upon removal of the Initialize Mode) and the check bits generated correspond to the all-zero data.

Table 4. 16-Bit Modified Hamming Code

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)		X	X	X		X			X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X

NOTE:
The check bit is generated as either an XOR or XNOR of the eight data bits noted by an "X" in the table.

Table 5. Syndrome Decode to Bit-In-Error

SYNDROME BITS											
			S8	0	1	0	1	0	1	0	1
			S4	0	0	1	1	0	0	1	1
			S2	0	0	0	0	1	1	1	1
SX	S0	S1									
0	0	0		*	C8	C4	T	C2	T	T	M
0	0	1		C1	T	T	15	T	13	7	T
0	1	0		C0	T	T	M	T	12	6	T
0	1	1		T	10	4	T	0	T	T	M
1	0	0		CX	T	T	14	T	11	5	T
1	0	1		T	9	3	T	M	T	T	M
1	1	0		T	8	2	T	1	T	T	M
1	1	1		M	T	T	M	T	M	M	T

* — no errors detected
Number — the location of the single bit-in-error
T — two errors detected
M — three or more errors detected

Table 6. Diagnostic Latch Loading

DATA BIT	INTERNAL FUNCTION
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6, 7	Don't Care
8	CODE ID 0
9	CODE ID 1
10	CODE ID 2
11	DIAG MODE 0
12	DIAG MODE 1
13	CORRECT
14	PASS THRU
15	Don't Care

Table 7. 32-Bit Modified Hamming Code

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CS	Even (XOR)	X				X		X	X	X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)	X	X	X	X	X	X	X	X								

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CS	Even (XOR)		X	X	X		X					X		X	X		X
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)									X	X	X	X	X	X	X	X

NOTE:
The check bit is generated as either an XOR or XNOR of the sixteen data bits noted by an "X" in the table.

Table 8. Syndrome Decode to Bit-In-Error

SYNDROME BITS				S16	0	1	0	1	0	1	0	1
				S8	0	0	1	1	0	0	1	1
				S4	0	0	0	0	1	1	1	1
SX	S0	S1	S2									
0	0	0	0		*	C16	C8	T	C4	T	T	30
0	0	0	1		C2	T	T	27	T	5	M	T
0	0	1	0		C1	T	T	25	T	3	15	T
0	0	1	1		T	M	13	T	23	T	T	M
0	1	0	0		C0	T	T	24	T	2	M	T
0	1	0	1		T	1	12	T	22	T	T	M
0	1	1	0		T	M	10	T	20	T	T	M
0	1	1	1		16	T	T	M	T	M	M	T
1	0	0	0		CX	T	T	M	T	M	14	T
1	0	0	1		T	M	11	T	21	T	T	M
1	0	1	0		T	M	9	T	19	T	T	31
1	0	1	1		M	T	T	29	T	7	M	T
1	1	0	0		T	M	8	T	18	T	T	M
1	1	0	1		17	T	T	28	T	6	M	T
1	1	1	0		M	T	T	26	T	4	M	T
1	1	1	1		T	0	M	T	M	T	T	M

* — no errors detected
Number — the location of the single bit-in-error
T — two errors detected
M — three or more errors detected

Table 9. Diagnostic Latch Loading

DATA BIT	INTERNAL FUNCTION
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6	Diagnostic Check Bit 16
7	Don't Care
8	Slice 0/1 — CODE ID 0
9	Slice 0/1 — CODE ID 1
10	Slice 0/1 — CODE ID 2
11	Slice 0/1 — DIAG MODE 0
12	Slice 0/1 — DIAG MODE 1
13	Slice 0/1 — CORRECT
14	Slice 0/1 — PASS THRU
15	Don't Care
16–23	Don't Care
24	Slice 2/3 — CODE ID 0
25	Slice 2/3 — CODE ID 1
26	Slice 2/3 — CODE ID 2
27	Slice 2/3 — DIAG MODE 0
28	Slice 2/3 — DIAG MODE 1
29	Slice 2/3 — CORRECT
30	Slice 2/3 — PASS THRU
31	Don't Care

Table 10. Key AC Calculations for the 32-Bit Configuration

32-BIT PROPAGATION DELAY		COMPONENT DELAY FROM 2960 AC SPECIFICATIONS, TABLE C
FROM	TO	
DATA	Check Bits Out	(DATA to SC) + (CB to SC, CODE ID 011)
DATA In	Corrected DATA Out	(DATA to SC) + (CB to SC, CODE ID 011) + (CB to DATA, CODE ID 010)
DATA	Syndromes Out	(DATA to SC) + (CB to SC, CODE ID 011)
DATA	$\overline{\text{ERROR}}$ for 32 Bits	(DATA to SC) + (CB to $\overline{\text{ERROR}}$, CODE ID 011)
DATA	$\overline{\text{MULT ERROR}}$ for 32 Bits	(DATA to SC) + (CB to $\overline{\text{MULT ERROR}}$, CODE ID 011)

Table 11. 64-Bit Modified Hamming Code Check Bit Encoding

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CX	Even (XOR)		X	X	X		X			X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)	X	X	X	X	X	X	X	X								
C32	Even (XOR)	X	X	X	X	X	X	X	X								

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
CX	Even (XOR)		X	X	X		X			X	X		X			X	
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)									X	X	X	X	X	X	X	X
C32	Even (XOR)									X	X	X	X	X	X	X	X

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
CX	Even (XOR)	X				X		X	X			X		X	X		X
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)	X	X	X	X	X	X	X	X								
C32	Even (XOR)									X	X	X	X	X	X	X	X

GENERATED CHECK BITS	PARITY	PARTICIPATING DATA BITS															
		48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
CX	Even (XOR)	X				X		X	X			X		X	X		X
C0	Even (XOR)	X	X	X		X		X		X		X		X			
C1	Odd (XNOR)	X			X	X			X		X	X			X		X
C2	Odd (XNOR)	X	X				X	X	X				X	X	X		
C4	Even (XOR)			X	X	X	X	X	X							X	X
C8	Even (XOR)									X	X	X	X	X	X	X	X
C16	Even (XOR)									X	X	X	X	X	X	X	X
C32	Even (XOR)	X	X	X	X	X	X	X	X								

NOTE:
The check bit is generated as either an XOR or XNOR of the 32 data bits noted by an "X" in the table.

Table 12. Syndrome Decode to Bit-In-Error

SYNDROME BITS																			
				S32	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
				S16	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
				S8	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
				S4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
SX	S0	S1	S2																	
0	0	0	0		*	C32	C16	T	C8	T	T	M	C4	T	T	M	T	46	62	T
0	0	0	1		C2	T	T	M	T	43	59	T	T	53	37	T	M	T	T	M
0	0	1	0		C1	T	T	M	T	41	57	T	T	51	35	T	15	T	T	31
0	0	1	1		T	M	M	T	13	T	T	29	23	T	T	7	T	M	M	T
0	1	0	0		C0	T	T	M	T	40	56	T	T	50	34	T	M	T	T	M
0	1	0	1		T	49	33	T	12	T	T	28	22	T	T	6	T	M	M	T
0	1	1	0		T	M	M	T	10	T	T	26	20	T	T	4	T	M	M	T
0	1	1	1		16	T	T	0	T	M	M	T	T	M	M	T	M	T	T	M
1	0	0	0		CX	T	T	M	T	M	M	T	T	M	M	T	14	T	T	30
1	0	0	1		T	M	M	T	11	T	T	27	21	T	T	5	T	M	M	T
1	0	1	0		T	M	M	T	9	T	T	25	19	T	T	3	T	47	63	T
1	0	1	1		M	T	T	M	T	45	61	T	T	55	39	T	M	T	T	M
1	1	0	0		T	M	M	T	8	T	T	24	18	T	T	2	T	M	M	T
1	1	0	1		17	T	T	1	T	44	60	T	T	54	38	T	M	T	T	M
1	1	1	0		M	T	T	M	T	42	58	T	T	52	36	T	M	T	T	M
1	1	1	1		T	48	32	T	M	T	T	M	M	T	T	M	T	M	M	T

* — no errors detected
Number — the location of the single bit-in-error
T — two errors detected
M — three or more errors detected

Table 13. Diagnostic Latch Loading

DATA BIT	INTERNAL FUNCTION
0	Diagnostic Check Bit X
1	Diagnostic Check Bit 0
2	Diagnostic Check Bit 1
3	Diagnostic Check Bit 2
4	Diagnostic Check Bit 4
5	Diagnostic Check Bit 8
6, 7	Don't Care
8	Slice 0/1 — CODE ID 0
9	Slice 0/1 — CODE ID 1
10	Slice 0/1 — CODE ID 2
11	Slice 0/1 — DIAG MODE 0
12	Slice 0/1 — DIAG MODE 1
13	Slice 0/1 — CORRECT
14	Slice 0/1 — PASS THRU
15	Don't Care
16-23	Don't Care
24	Slice 2/3 — CODE ID 0
25	Slice 2/3 — CODE ID 1
26	Slice 2/3 — CODE ID 2
27	Slice 2/3 — DIAG MODE 0
28	Slice 2/3 — DIAG MODE 1
29	Slice 2/3 — CORRECT
30	Slice 2/3 — PASS THRU

DATA BIT	INTERNAL FUNCTION
31	Don't Care
32-37	Don't Care
38	Diagnostic Check Bit 16
39	Don't Care
40	Slice 4/5 — CODE ID 0
41	Slice 4/5 — CODE ID 1
42	Slice 4/5 — CODE ID 2
43	Slice 4/5 — DIAG MODE 0
44	Slice 4/5 — DIAG MODE 1
45	Slice 4/5 — CORRECT
46	Slice 4/5 — PASS THRU
47	Don't Care
48-54	Don't Care
55	Diagnostic Check Bit 32
56	Slice 6/7 — CODE ID 0
57	Slice 6/7 — CODE ID 1
58	Slice 6/7 — CODE ID 2
59	Slice 6/7 — DIAG MODE 0
60	Slice 6/7 — DIAG MODE 1
61	Slice 6/7 — CORRECT
62	Slice 6/7 — PASS THRU
63	Don't Care

Table 14. Key AC Calculations for the 64-Bit Configuration

64-BIT PROPAGATION DELAY		COMPONENT DELAYS FROM 2960 AC SPECIFICATIONS, TABLE C (PLUS MSI)
FROM	TO	
DATA	Check Bits Out	(DATA to SC) + (XOR Delay)
DATA In	Corrected DATA Out	(DATA to SC) + (XOR Delay) + (Buffer Delay) + (CB to DATA, CODE ID 1xx)
DATA	Syndromes	(DATA to SC) + (XOR Delay)
DATA	$\overline{\text{ERROR}}$ for 64 Bits	(DATA to SC) + (XOR Delay) + (NOR Delay)
DATA	$\overline{\text{MULT ERROR}}$ for 64 Bits	(DATA to SC) + (XOR Delay) + (Buffer Delay) + (CB to $\overline{\text{MULT ERROR}}$, CODE ID 1xx)
DATA	DOUBLE ERROR for 64 Bits	(DATA to SC) + (XOR Delay) + (XOR/NOR Delay)

Table 15. TOME (Three or More Errors)*

			S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
			**S6	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
			S5	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
			S4	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S1	S2	S3																	
0	0	0		0	0	0	1	0	1	1	1	0	1	1	1	0	0	0	0
0	0	1		0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
0	1	0		0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1		1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
1	0	0		0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
1	0	1		0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
1	1	0		1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1
1	1	1		0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1

*S6, S5, . . . S0 are internal syndromes except in Modes 010, 100, 101, 110, 111, (CODE ID_2, ID_1, ID_0). In these modes the syndromes are input over the Check-Bit lines S6 — C6, S5 — C5, S1 — C1, S0 — C0.

**The S6 internal syndrome is always forced to 0 in CODE ID 000.

Table 16. Syndrome/Check Bit Generation in GENERATE Mode

GENERATE MODE (CHECK BITS) \ CODE ID_{2-0}	000	010	011	100	101	110	111
SC_0 ·	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_3$	$PG_2 \oplus PG_4$	$PG_2 \oplus PG_3$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_4$	$PG_1 \oplus PG_4$
SC_1 ·	PA	PA	PA	PA	PA	PA	PA
SC_2 ·	$\overline{PD}$	$\overline{PD}$	PD	$\overline{PD}$	PD	PD	PD
SC_3 ·	$\overline{PE}$	$\overline{PE}$	PE	$\overline{PE}$	PE	PE	PE
SC_4 ·	PF	PF	PF	PF	PF	PF	PF
SC_5 ·	PC	PC	PC	PC	PC	PC	PC
SC_6 ·	1	PB	PC	PB	PB	PB	PB

Table 17. Syndrome/Check Bit Generation in Detect/Correct Modes

DETECT AND CORRECT MODES (SYNDROMES) \ CODE ID_{2-0}	000	010	011*	100	101	110	111
SC_0 ·	$PG_2 \oplus PG_3 \oplus C0$	$PG_1 \oplus PG_3 \oplus C0$	$PG_2 \oplus PG_4 \oplus CB_0$	$PG_2 \oplus PG_3 \oplus C0$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_4$	$PG_1 \oplus PG_4$
SC_1 ·	$PA \oplus C1$	$PA \oplus C1$	$PA \oplus CB_1$	$PA \oplus C1$	PA	PA	PA
SC_2 ·	$\overline{PD} \oplus C2$	$\overline{PD} \oplus C2$	$PD \oplus CB_2$	$\overline{PD} \oplus C2$	PD	PD	PD
SC_3 ·	$\overline{PE} \oplus C3$	$\overline{PE} \oplus C3$	$PE \oplus CB_3$	$\overline{PE} \oplus C3$	PE	PE	PE
SC_4 ·	$PF \oplus C4$	$PF \oplus C4$	$PF \oplus CB_4$	$PF \oplus C4$	PF	PF	PF
SC_5 ·	$PC \oplus C5$	$PC \oplus C5$	$PC \oplus CB_5$	$PC \oplus C5$	PC	PC	PC
SC_6 ·	1	$PB \oplus C6$	$PC \oplus CB_6$	PB	PB	$PB \oplus C6$	$PB \oplus C6$

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent, the Data Latch operates normally.

Table 18. Syndrome/Check Bit Generation in Diagnostic Read Mode

DIAGNOSTIC READ MODE \ CODE ID_{2-0}	000	010	011*	100	101	110	111
SC_0 ·	$PG_2 \oplus PG_3 \oplus DL_0$	$PG_1 \oplus PG_3 \oplus DL_0$	$PG_2 \oplus PG_4 \oplus CB_0$	$PG_2 \oplus PG_3 \oplus DL_0$	$PG_2 \oplus PG_3$	$PG_1 \oplus PG_4$	$PG_1 \oplus PG_4$
SC_1 ·	$PA \oplus DL_1$	$PA \oplus DL_1$	$PA \oplus CB_1$	$PA \oplus DL_1$	PA	PA	PA
SC_2 ·	$\overline{PD} \oplus DL_2$	$\overline{PD} \oplus DL_2$	$PD \oplus CB_2$	$\overline{PD} \oplus DL_2$	PD	PD	PD
SC_3 ·	$\overline{PE} \oplus DL_3$	$\overline{PE} \oplus DL_3$	$PE \oplus CB_3$	$\overline{PE} \oplus DL_3$	PE	PE	PE
SC_4 ·	$PF \oplus DL_4$	$PF \oplus DL_4$	$PF \oplus CB_4$	$PF \oplus DL_4$	PF	PF	PF
SC_5 ·	$PC \oplus DL_5$	$PC \oplus DL_5$	$PC \oplus CB_5$	$PC \oplus DL_5$	PC	PC	PC
SC_6 ·	1	$PB \oplus DL_6$	$PC \oplus CB_6$	PB	PB	$PB \oplus DL_6$	$PB \oplus DL_6$

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent, the Data Latch operates normally.

Table 19. Syndrome/Check Bit Generation in Diagnostic Write Mode

DIAGNOSTIC WRITE MODE \ CODE ID_{2-0}	000	010	011*	100	101	110	111
SC_0 ←	DL_0	DL_0	CB_0	DL_0	1	1	1
SC_1 ←	DL_1	DL_1	CB_1	DL_1	1	1	1
SC_2 ←	DL_2	DL_2	CB_2	DL_2	1	1	1
SC_3 ←	DL_3	DL_3	CB_3	DL_3	1	1	1
SC_4 ←	DL_4	DL_4	CB_4	DL_4	1	1	1
SC_5 ←	DL_5	DL_5	CB_5	DL_5	1	1	1
SC_6 ←	1	DL_6	CB_6	1	1	DL_6	DL_7

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent; the Data Input Latch operates normally.

Table 20. Syndrome/Check Bit Generation in PASS THRU Mode

PASS THRU MODE \ CODE ID_{2-0}	000	010	011*	100	101	110	111
SC_0 ←	C0	C0	CB_0	C0	1	1	1
SC_1 ←	C1	C1	CB_1	C1	1	1	1
SC_2 ←	C2	C2	CB_2	C2	1	1	1
SC_3 ←	C3	C3	CB_3	C3	1	1	1
SC_4 ←	C4	C4	CB_4	C4	1	1	1
SC_5 ←	C5	C5	CB_5	C5	1	1	1
SC_6 ←	1	C6	CB_6	1	1	C6	C6

*In CODE ID_{2-0} 011 the Check-Bit Latch is forced transparent; the Data Input Latch operates normally.

Table 21. CODE $ID_{2-0} = 000$*

	S5	0	0	0	0	1	1	1	1
	S4	0	0	1	1	0	0	1	1
	S3	0	1	0	1	0	1	0	1
S2	S1								
0	0	—	—	—	5	—	11	14	—
0	1	—	1	2	6	8	12	—	—
1	0	—	—	3	7	9	13	15	—
1	1	—	0	4	—	10	—	—	—

*Unlisted S combinations are no correction.

Table 22. CODE $ID_{2-0} = 010$*

	CB_6	0	0	0	0	1	1	1	1
	CB_5	1	1	1	1	0	0	0	0
	CB_4	0	0	1	1	0	0	1	1
	CB_3	0	1	0	1	0	1	0	1
CB_2	CB_1								
0	0	—	11	14	—	—	—	—	5
0	1	8	12	—	—	—	1	2	6
1	0	9	13	15	—	—	—	3	7
1	1	10	—	—	—	—	0	4	—

*Unlisted CB combinations are no correction.

Table 23. CODE $ID_{2-0} = 011$*

	S6	0	0	0	0	1	1	1	1
	S5	0	0	0	0	1	1	1	1
	S4	0	0	1	1	0	0	1	1
	S3	0	1	0	1	0	1	0	1
S2	S1								
0	0	—	—	—	5	—	11	14	—
0	1	—	1	2	6	8	12	—	—
1	0	—	—	3	7	9	13	15	—
1	1	—	0	4	—	10	—	—	—

*Unlisted S combinations are no correction.

Table 24. CODE $ID_{2-0} = 100$*

	CB_0	0	0	0	0	1	1	1	1
	CB_6	0	0	0	0	1	1	1	1
	CB_5	1	1	1	1	0	0	0	0
	CB_4	0	0	1	1	0	0	1	1
	CB_3	0	1	0	1	0	1	0	1
CB_2	CB_1								
0	0	—	11	14	—	—	—	—	5
0	1	8	12	—	—	—	1	2	6
1	0	9	13	15	—	—	—	3	7
1	1	10	—	—	—	—	0	4	—

*Unlisted CB combinations are no correction.

Table 25. CODE ID_{2-0} = 101*

	CB_0	0	0	0	0	1	1	1	1
	CB_6	0	0	0	0	1	1	1	1
	CB_5	0	0	0	0	1	1	1	1
	CB_4	0	0	1	1	0	0	1	1
	CB_3	0	1	0	1	0	1	0	1
CB_2	CB_1								
0	0	—	—	—	5	—	11	14	—
0	1	—	1	2	6	8	12	—	—
1	0	—	—	3	7	9	13	15	—
1	1	—	0	4	—	10	—	—	—

*Unlisted CB combinations are no correction.

Table 26. CODE ID_{2-0} = 110*

	CB_0	0	0	0	0	1	1	1	1
	CB_6	1	1	1	1	0	0	0	0
	CB_5	0	0	0	0	1	1	1	1
	CB_4	0	0	1	1	0	0	1	1
	CB_3	0	1	0	1	0	1	0	1
CB_2	CB_1								
0	0	—	—	—	5	—	11	14	—
0	1	—	1	2	6	8	12	—	—
1	0	—	—	3	7	9	13	15	—
1	1	—	0	4	—	10	—	—	—

*Unlisted CB combinations are no correction.

Table 27. CODE ID_{2-0} = 111*

	CB_0	0	0	0	0	1	1	1	1
	CB_6	1	1	1	1	0	0	0	0
	CB_5	1	1	1	1	0	0	0	0
	CB_4	0	0	1	1	0	0	1	1
	CB_3	0	1	0	1	0	1	0	1
CB_2	CB_1								
0	0	—	11	14	—	—	—	—	5
0	1	8	12	—	—	—	1	2	6
1	0	9	13	15	—	—	—	3	7
1	1	10	—	—	—	—	0	4	—

*Unlisted CB combinations are no correction.

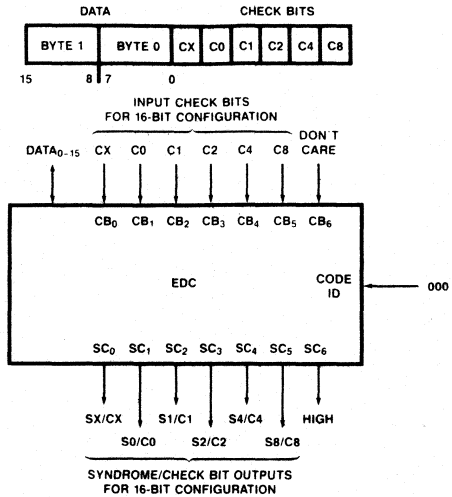


Figure 2. 16-Bit Data Format and I/O

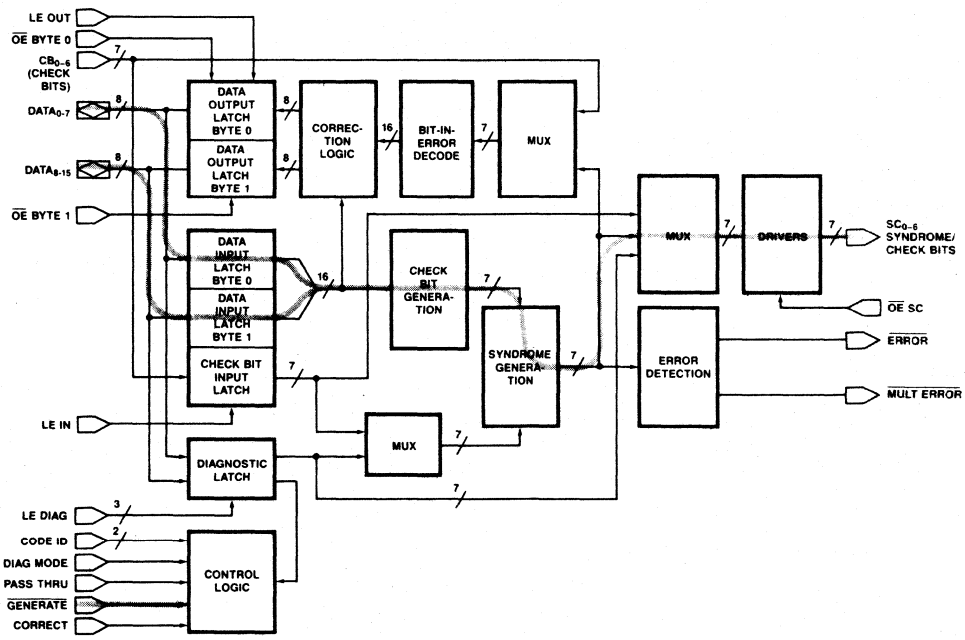


Figure 3. Check Bit Generation

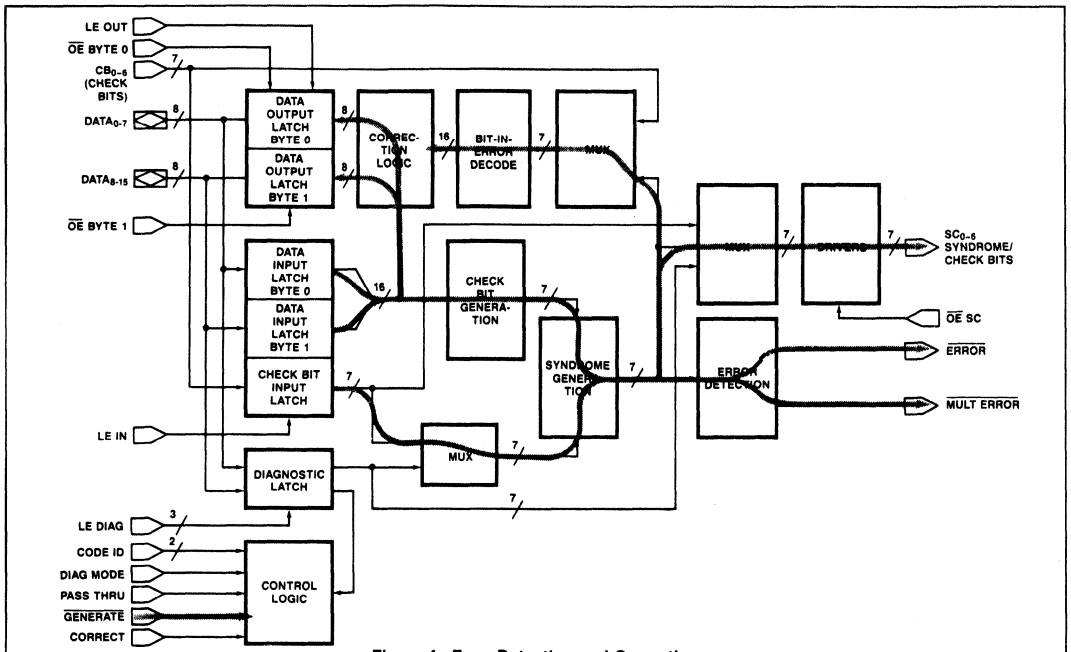


Figure 4. Error Detection and Correction

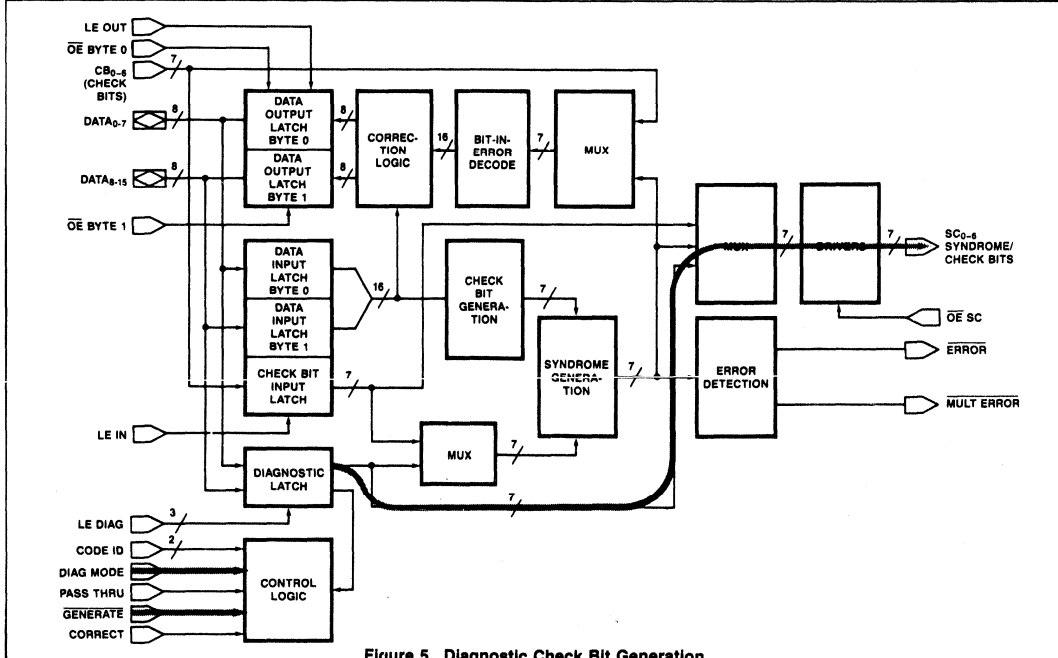


Figure 5. Diagnostic Check Bit Generation

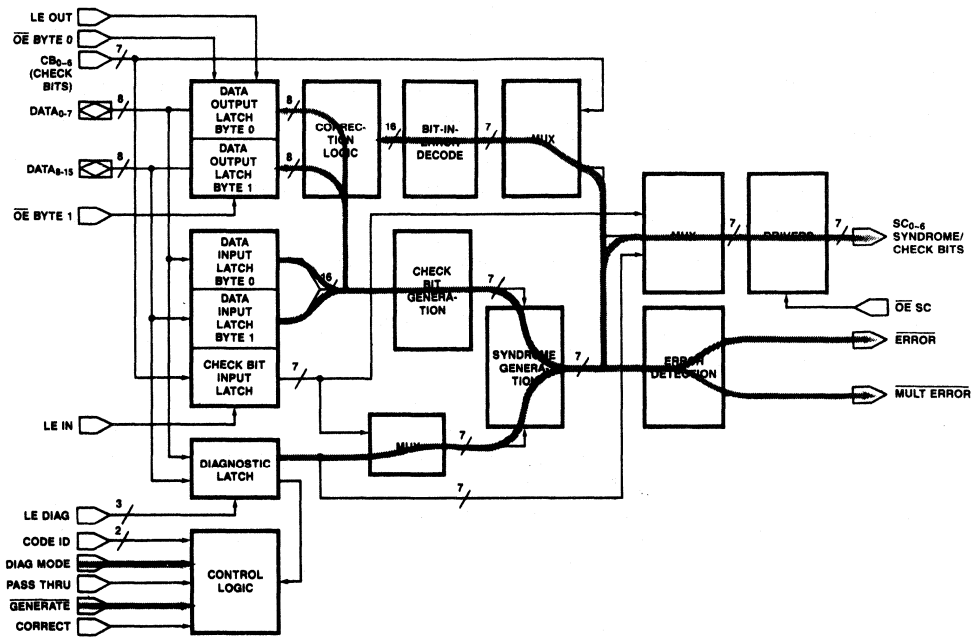
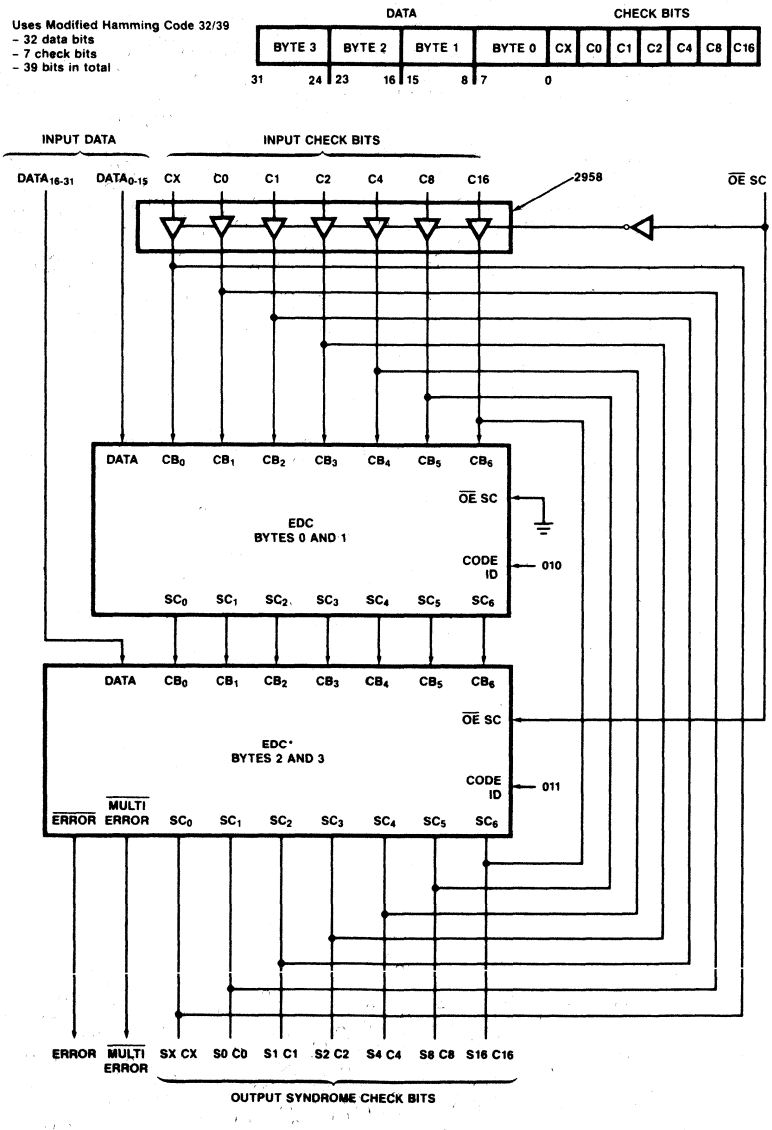


Figure 6. Diagnostic Detect and Correct



*Check Bit Latch is forced transparent in this CODE ID combination for this slice.

Figure 7. 32-Bit Data Format and I/O

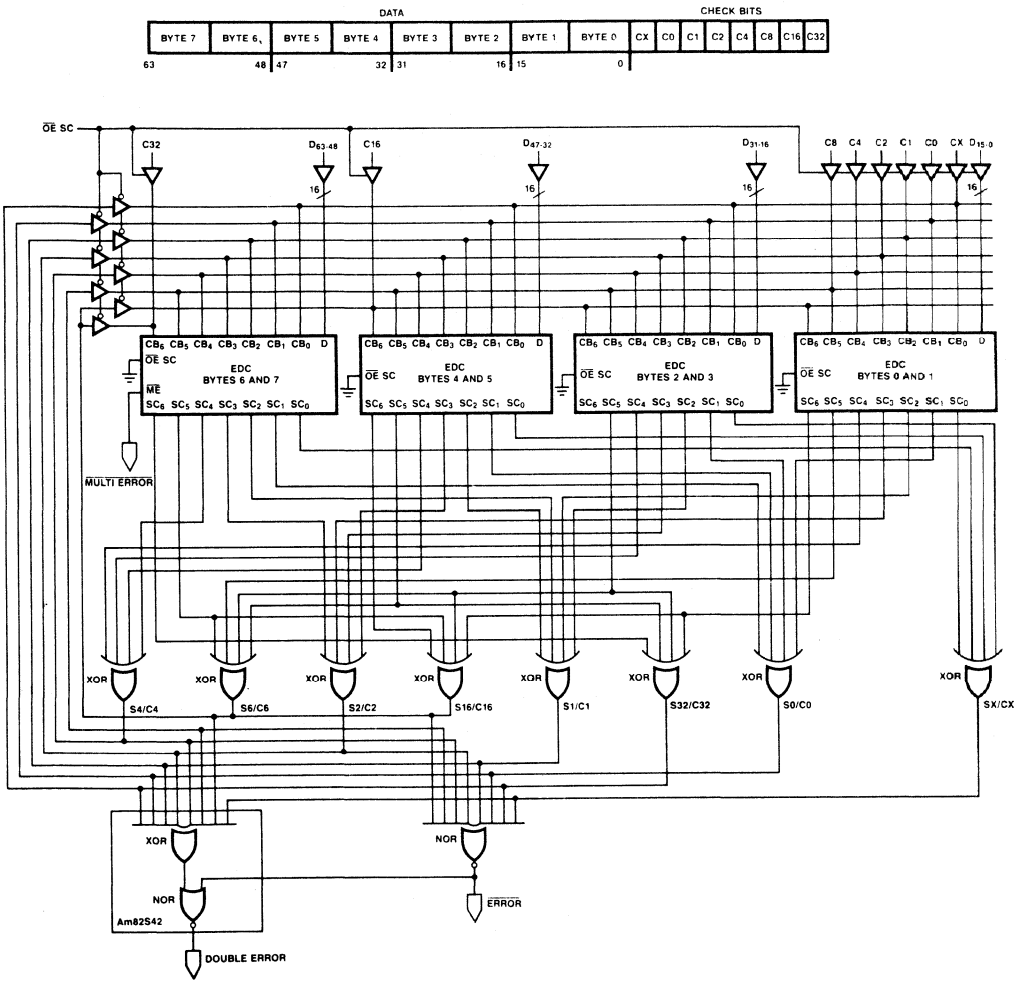


Figure 8. 64-Bit Data Format and I/O Configuration

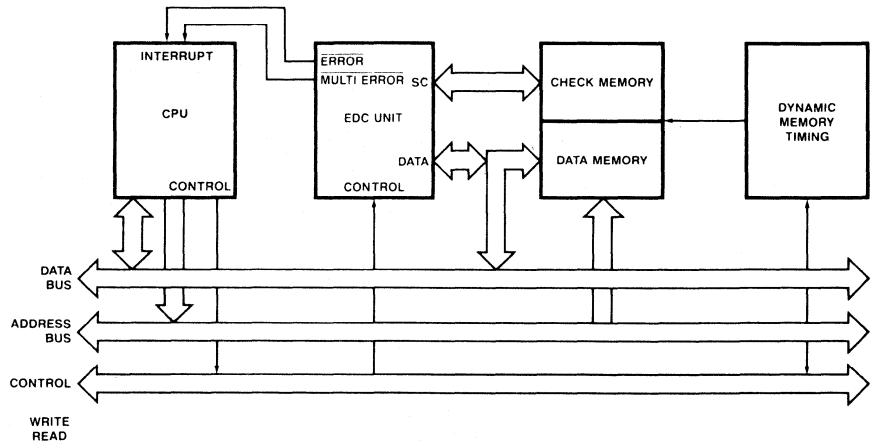


Figure 9. Check-Only Configuration

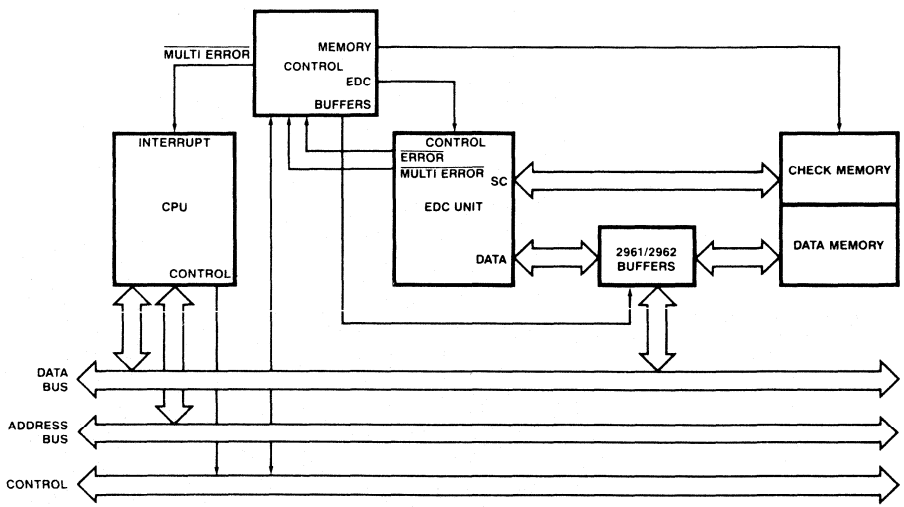


Figure 10. Correct-Always Configuration

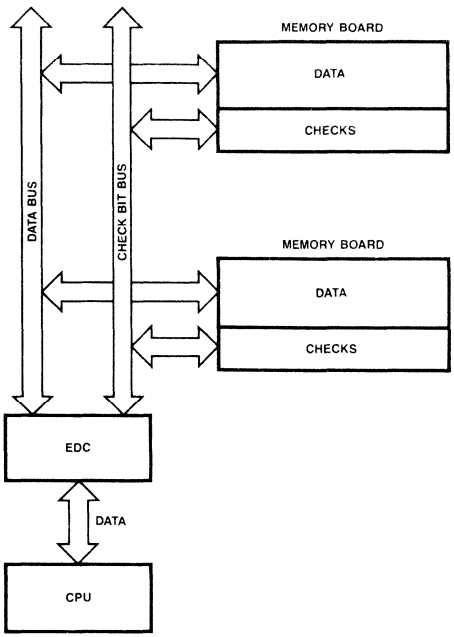


Figure 11. EDC Per System

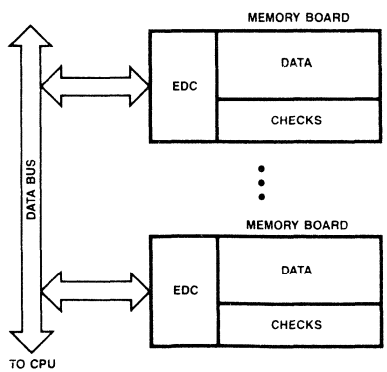


Figure 12. EDC Per Board

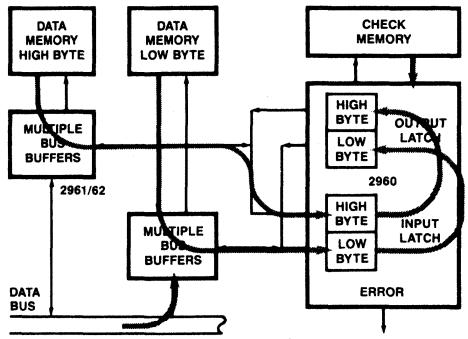


Figure 13. Byte Write, Phase 1: Read Out the Old Word and Correct

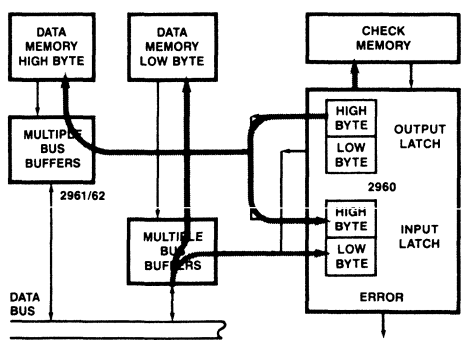


Figure 14. Byte Write, Phase 2: Insert the New Byte Generate Checks and Write Into Memory

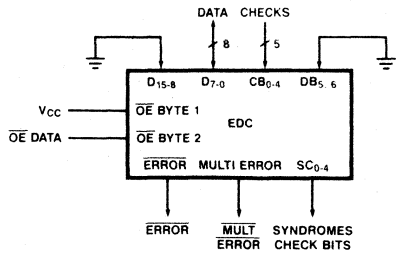


Figure 15. 8-Bit Configuration

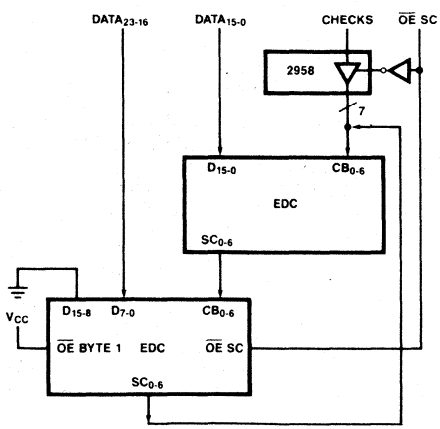
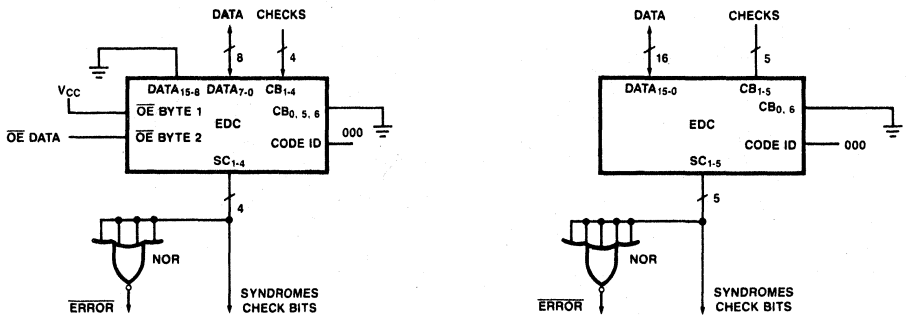
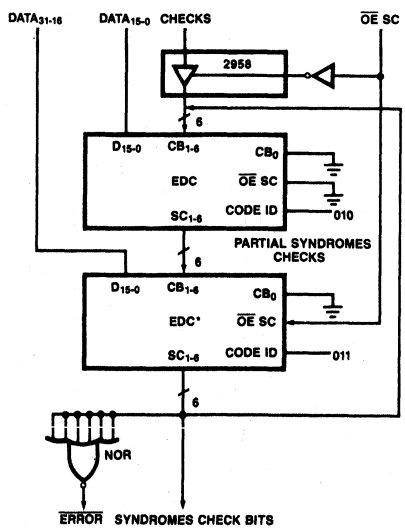


Figure 16. 24-Bit Configuration



a. 8-Bit Data Word

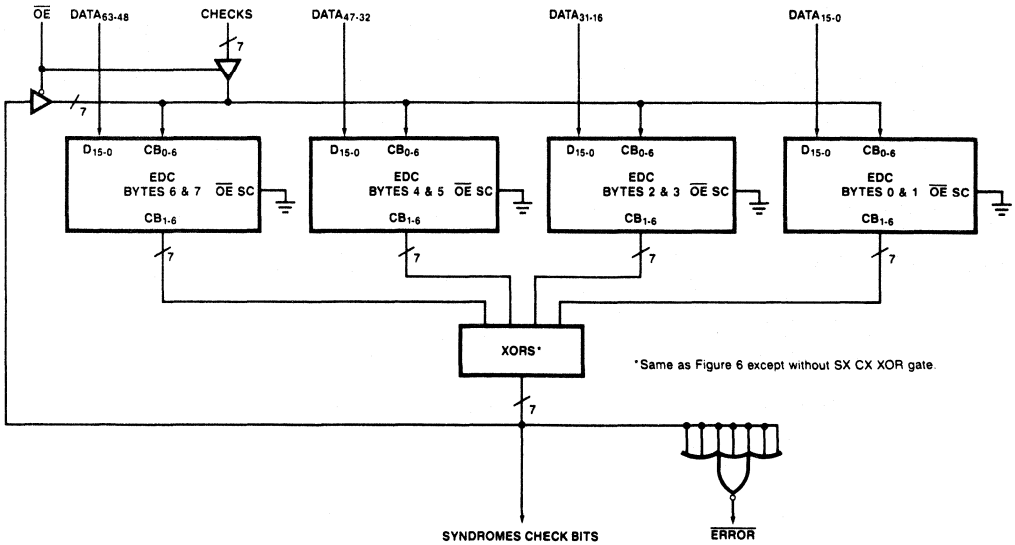
b. 16-Bit Data Word



*The code ID combination for this slice forces the check bit latch transparent.

c. 32-Bit Data Word

Figure 17. Single Error "Correction Only" Configurations



NOTES:
1. In Pass Thru Mode the contents of the check latch appear on the XOR outputs inverted.
2. In Diagnostic GENERATE Mode the contents of the diagnostic latch appear on the XOR outputs inverted.

d. 64-Bit Data Word

Figure 17. (Continued)

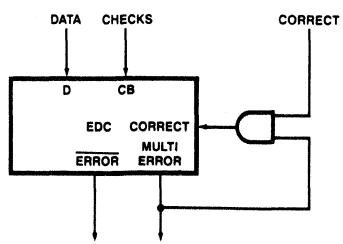


Figure 18. Inhibition of Data Modification

DYNAMIC MEMORY CONTROLLER

Originally published by Signetics January 1984

FEATURES

- **Operating Flexibility—controls 16K or 64K dynamic RAMSs**
- **8-Bit Refresh Counter—refresh address generation, clear input, and selectable terminal count (128 or 256) output**
- **Row Address Decoder—four active Row Address Select (RAS) outputs during refresh**
- **On-Chip Latches—dual 8-bit address latches and RAS decoder latches**
- **User-Selectable Refresh Modes—burst, distributed, or transparent**
- **3-port, 8-bit address multiplexer with Schottky speed**
- **Non-inverting address for RAS and CAS signal paths**

PRODUCT DESCRIPTION

The Signetics 2964 Dynamic Memory Controller (DMC) provides address multiplexing, refresh address generation, and Row/Column control for MOS dynamic memories of any data width. The eight bit address path is designed for 64K RAMs but can be used equally well with 16K RAMs. Sixteen address input latches and two row address select latches (for higher order address) allow the DMC to control up to 256K words of memory (with 64K RAMs) by using the internal row address decoder to select from one-of-four banks of RAMs.

FUNCTIONAL OPERATION

The Signetics 2964B Dynamic Memory Controller (Figure 1) replaces a dozen MSI devices by grouping several unique functions. Two 8-bit latches capture and hold the memory address. These latches and a clearable, 8-bit refresh counter feed into an 8-bit, 3-input, Schottky speed MUX, for output to the DRAM address lines.

The 2964B also includes a special RAS decoder and CAS buffer. Placing these functions on the same chip minimizes the time skew between output functions which would otherwise be separate MSI chips, and therefore, allows a faster memory cycle time by the amount of skew eliminated.

The RAS Decoder allows upper addresses to select one-of-four banks of DRAM by determining which bank receives a RAS input. During refresh ($\overline{RFSH}$ = LOW), the decoder mode is changed to four-of-four and all banks of memory receive a RAS input for refresh in response to a RASI active LOW input. CAS is inhibited during refresh.

Burst mode refresh is accomplished by holding RFSH low and toggling RASI.

A_{15} is a dual function input which controls the refresh counter's range. For 64K DRAMs, it is an address input. For 16K DRAMs, it can be pulled to + 12V through 1K to terminate the refresh count at 128 instead of 256.

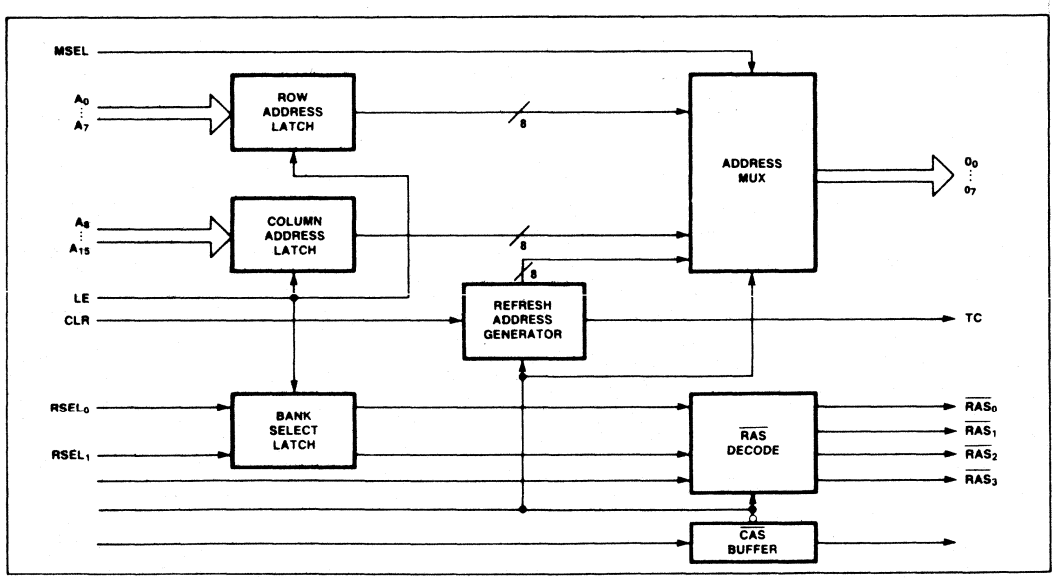


Figure 1. Block Diagram of 2964B Dynamic Memory Controller

DATA COMMUNICATIONS

Universal Asynchronous Receiver/Transmitter (UART)

Advance Information

Originally published by Signetics February 1985

DESCRIPTION

The Signetics SCC2691 Universal Asynchronous Receiver/Transmitter (UART) is a single chip CMOS-LSI communications device that provides a full-duplex asynchronous receiver/transmitter in a single 24 pin DIP. It is fabricated with Signetics CMOS technology which combines the benefits of high density and low power consumption.

The operating speed of the receiver and transmitter can be selected independently as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the UART particularly attractive for dual-speed channel applications such as clustered terminal systems.

The receiver is quadruply buffered to minimize the potential of receiver overrun or to reduce interrupt overhead in interrupt driven systems. In addition, a handshaking capability is provided to disable a remote UART transmitter when the receiver buffer is full.

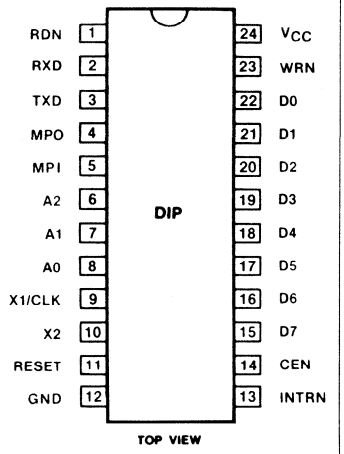
The UART provides a power down mode in which the oscillator is frozen but the register contents are stored. This results in reduced power consumption on the order of several magnitudes.

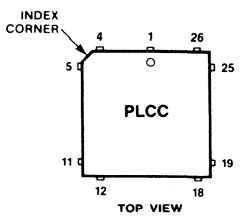
The UART is fully TTL compatible and operates from a single +5V power supply.

FEATURES

- **Full-duplex asynchronous receiver/transmitter**
- **Quadruple buffered receiver data register**
- **Programmable data format:**
  - **5 to 8 data bits plus parity**
  - **Odd, even, no parity or force parity**
  - **I, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Baud rate for the receiver and transmitter selectable from:**
  - **18 fixed rates: 50 to 38.4K baud**
  - **One user defined rate derived from programmable timer/counter**
  - **External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **Normal (full-duplex)**
  - **Automatic echo**
  - **Local loopback**
  - **Remote loopback**
- **Multi-function programmable 16-bit counter/timer**
- **Single interrupt output with seven maskable interrupting conditions**
- **On-chip crystal oscillator**
- **Low power mode**
- **TTL compatible**
- **Single +5V power supply**
- **24-pin, 300 mil c-c DIP**

PIN CONFIGURATION





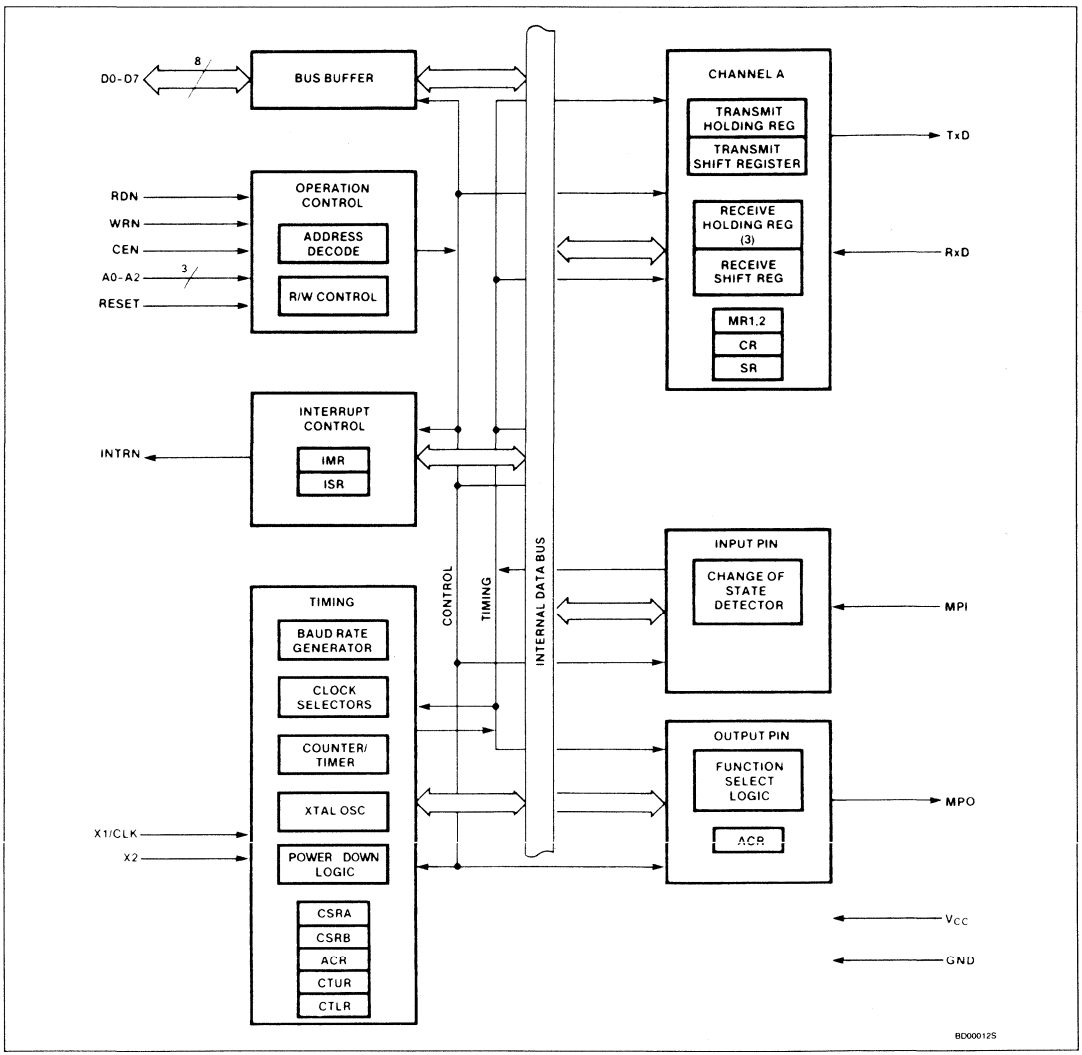
Pin	Function	Pin	Function
1	NC	15	NC
2	RDN	16	INTRN
3	RXD	17	CEN
4	TXD	18	D7
5	MPO	19	D6
6	MPI	20	D5
7	NC	21	D4
8	A2	22	D3
9	A1	23	NC
10	A0	24	D2
11	X1/CLK	25	D1
12	X2	26	D0
13	RESET	27	WRN
14	GND	28	V_{CC}

Advance Information

ORDERING CODE

PACKAGES	V_{CC}=5V±10%, T_A=0°C to 70°C
Plastic DIP	SCC2691AC1N24
Plastic LCC	SCC2691C1A28

BLOCK DIAGRAM



Advance Information

PIN DESCRIPTION

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
D0 – D7	22 – 15	26 – 24 22 – 18	I/O	**Data Bus:** Active high 8-bit bidirectional three-state data bus. Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the UART take place over this bus. The direction of the transfer is controlled by the WRN and RDN inputs when the CEN input is low. When the CEN input is high, the data bus is in the three-state condition.
CEN	14	17	I	**Chip Enable:** Active low input. When low, data transfers between the CPU and the UART are enabled on D0 – D7 as controlled by the WRN, RDN, and A0 – A2 inputs. When CEN is high, the UART is effectively isolated from the data bus and D0 – D7 are placed in the three-state condition.
WRN	23	27	I	**Write Strobe:** Active low input. A low on this pin while CEN is low causes the contents of the data bus to be transferred to the register selected by A0 – A2. The transfer occurs on the trailing (rising) edge of the signal.
RDN	1	2	I	**Read Strobe:** Active low input. A low on this pin while CEN is low causes the contents of the register selected by A0 – A2 to be placed on the data bus. The read cycle begins on the leading (falling) edge of RDN.
A0 – A2	8 – 6	10 – 8	I	**Address Inputs:** Active high address inputs to select the UART registers for read/write operations.
RESET	11	13	I	**Reset:** Master reset. A high on this pin clears the status register (SR), clears the interrupt mask register (IMR), clears the auxiliary control register (ACR), places the receiver and transmitter in the inactive state causing the TXD output to go to the marking (high) state, and stops the counter/timer.
INTRN	13	16	O	**Interrupt Request:** This active low output is asserted upon occurrence of one or more of seven maskable interrupting conditions. The CPU can read the interrupt status register to determine the interrupting condition(s).
X1/CLK	9	11	I	**Crystal 1:** Crystal or external clock input. When using the crystal oscillator, this pin serves as the connection for one side of the crystal. If a crystal is not used, an external clock is supplied at this input. An external clock (or crystal) is required even if the internal baud rate generator is not utilized. This clock is used to drive the internal baud rate generator, as an optional input to the timer/counter, and to provide other clocking signals required by the chip.
X2	10	12	I	**Crystal 2:** Connection for other side of crystal. If an external source is used instead of a crystal, this connection should be open or connected as shown in figure 5.
RXD	2	3	I	**Receiver Serial Data Input:** The least significant bit is received first. If external receiver clock is specified, this input is sampled on the rising edge of the clock.
TXD	3	4	O	**Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the marking (high) condition when the transmitter is idle or disabled and when the UART is operating in local loopback mode. If external transmitter clock is specified, the data is shifted on the falling edge of the transmitter clock.
MPO	4	5	O	**Multi-Purpose Output:** One of the following functions can be selected for this output pin by programming the auxiliary control register: **RTSN** – Request to send active low output. This output is asserted and negated via the command register. By appropriate programming of the mode registers, RTSN can be programmed to be automatically reset after the character in the transmitter is completely shifted or when the receiver FIFO and shift register are full. **C/TO** – The counter/timer output. **TXC1X** – The 1X clock for the transmitter. **TXC16X** – The 16X clock for the transmitter. **RXC1X** – The 1X clock for the receiver. **RXC16X** – The 16X clock for the receiver. **TXRDY** – The transmitter holding register empty signal. Active low interrupt. **RXRDY/FFULL** – The receiver FIFO not empty/full signal. Active low interrupt.
MPI	5	6	I	**Multi-Purpose Input:** This pin can be programmed to serve as an input for one of the following functions: **GPI** – General purpose input. The current state of the pin can be determined by reading the ISR. **CTSN** – Clear-to-Send active low input. **CTCLK** – Counter/timer external clock input. **RTCLK** – Receiver and/or transmitter external clock input. This may be a 1X or 16X clock as programmed by CSR[3:0] or CSR[7:4].
V_{CC}	24	28	I	**Power Supply:** +5V supply input
GND	12	14	I	**Ground**

Advance Information

BLOCK DIAGRAM

As shown on the block diagram, the UART consists of: data bus buffer, interrupt control, operation control, timing, receiver and transmitter.

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data buses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the UART.

Interrupt Control

A single interrupt output (INTRN) is provided which is asserted upon the occurrence of any of the following internal events:

- Transmit holding register ready
- Transmit shift register empty
- Receive holding register ready or FIFO full
- Change in break received status
- Counter reached terminal count
- Change in MPI input
- High level at the MPI input

Associated with the interrupt system are the interrupt mask register (IMR) and the interrupt status register (ISR). The IMR can be programmed to select only certain of the above conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions. However, the bits of the ISR are not masked by the IMR.

Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The functions performed by the CPU read and write operations are shown in table 1.

Table 1. REGISTER ADDRESSING

A2	A1	A0	READ (RDN=0)	WRITE (WRN=0)
0	0	0	MR1,MR2	MR1, MR2
0	0	1	SR	CSR
0	1	0	Reserved*	CR
0	1	1	RHR	THR
1	0	0	Reserved*	ACR
1	0	1	ISR	IMR
1	1	0	CTU	CTUR
1	1	1	(CTL)	CTLR

*Reserved registers should never be read during normal operation since they are reserved for internal diagnostics.

ACR = Auxiliary control register
CR = Command register
CSR = Clock select register
CTL = Counter/timer lower
CTLR = Counter/timer lower register
CTU = Counter/timer upper
CTUR = Couter/timer upper register
MR = Mode register A
SR = Status register
THR = TX holding register

Mode registers 1 and 2 are accessed via an auxiliary pointer. The pointer is set to MR1 by RESET or by issuing a reset pointer command via the command register. Any read or write of the mode register while the pointer is at MR1 switches the pointer to MR2. The pointer then remains at MR2 so that subsequent accesses are to MR2, unless the pointer is reset to MR1 as described above.

Timing Circuits

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and two clock selectors.

The crystal oscillator operates directly from a 3.6864MHz crystal connected across the X1/CLK and X2 inputs with a minimum of external components. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. If an external clock is used instead of a crystal, both X1 and X2 are driven using a configuration similar to the one in figure 5. Also, an arrangement where X2 is left open can be used, however, the input high voltage must be capable of attaining 4.4V. The clock serves as the basic timing reference for the baud rate generator (BRG), the counter/timer, and other internal circuits. A clock frequency, within the limits specified in the electrical specifications, must be supplied even if the internal BRG is not used.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4K baud. Thirteen of these are available simultaneously for use by the receiver and transmitter. Eight are fixed, and one of two sets of five can be selected by programming ACR[7]. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The clock selectors allow the independent selection by the receiver and transmitter of any of these baud rates or an external timing signal.

The C/T operation is programmed by ACR[6:4]. One of eight timing sources can be used as the input to the C/T. The output of the C/T is available to the clock selectors and can also be programmed by ACR[2:0], to be output on the MPO pin.

In the timer mode, the C/T generates a square wave whose period is twice the number of clock periods loaded into the C/T upper and lower registers. The counter ready bit in the ISR is set once each cycle of the square wave. If the value in CTUR or CTLR is changed, the current half period will not be affected, but subsequent half periods will be affected. In this mode the C/T runs continuously and does not recognize the stop counter command (the command only resets the counter ready bit in the ISR). Receipt of a start C/T command causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR. Counting begins upon receipt of a start C/T command. Upon reaching terminal count, the counter ready bit in the ISR is set. The counter continues counting past the terminal count until stopped by the CPU. If MPO is programmed to be the output of the C/T, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state and the counter ready bit is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command following a stop counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8 bits to the upper 8 bits occurs between the times that both halves of the counter are read. However a subsequent start counter command causes the counter to begin a new count cycle using the values in CTUR and CTLR.

Receiver and Transmitter

The UART is a full duplex asynchronous receiver/transmitter. The operating frequency for the receiver and transmitter can be selected independently from the baud rate generator, the counter timer, or from an external input. Registers associated with the communications channel are the mode registers (MR1 and MR2), the clock select register (CSR), the command register (CR), the status register (SR), the transmit holding register (THR), and the receive holding register (RHR).

Transmitter

The transmitter accepts parallel data from the CPU and converts it to a serial bit stream on

Advance Information

the TXD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TXD output remains high and the TXEMT bit in the SR will be set to 1. Transmission resumes and the TXEMT bit is cleared when the CPU loads a new character into the THR. In the 16X clock mode, this also resynchronizes the internal 1X transmitter clock so that transmission of the new character begins with minimum delay.

The transmitter can be forced to send a break (continuous low condition) by issuing a start break command via the CR. The break is terminated by a stop break command.

If the transmitter is disabled, it continues operating until the character currently being transmitted and the character in the THR, if any, are completely sent out. Characters cannot be loaded into the THR while the transmitter is disabled.

Receiver

The receiver accepts serial data on the RxD pin, converts the serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition, and presents the assembled character to the CPU. The receiver looks for a high to low (mark to space) transition of the start bit on the RXD input pin. If a transition is detected, the state of the RXD pin is sampled again each 16X clock for 7-1/2 clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RXD is sampled high, the start bit is invalid and the search for a valid start bit begins again. If RXD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least significant bit is received first. The data is then transferred to the RHR and the RXRDY bit in the SR is set to a 1. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (i.e. framing error) and RXD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one half bit time after the stop bit was sampled). The parity error, framing error and overrun error (if any) are strobed into the SR at the received character boundary, before the RXRDY status bit is set.

If a break condition is detected (RxD is low for the entire character including the stop bit) only one character consisting of all zeros will be loaded into the FIFO and the received break bit in the SR is set to 1. The RXD input must return to a high condition for two successive clock edges of the 1X clock (internal or external) before a search for the next start bit begins.

RECEIVER FIFO

The RHR consists of a first-in-first-out (FIFO) queue with a capacity of three characters. Data is loaded from the receive shift register into the top-most empty position of the FIFO. The RXRDY bit in the status register (SR) is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three queue positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits are 'popped' thus emptying a FIFO position for new data.

In addition to the data word, three status bits (parity error, framing error, and received break) are appended to each data character in the FIFO. Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the block mode, the status provided in the SR for these three bits is the logical OR of the status for all characters coming to the top of the FIFO since the last reset error command was issued. In either mode, reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore, the SR should be read prior to reading the corresponding data character.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available. If an additional character is received while this state exists, the contents of the FIFO are not affected: the character previously in the shift register is lost and the overrun error status bit (SR[4]) will be set upon receipt of the start bit of the new (overrunning) character.

WAKE UP MODE

In addition to the normal transmitter and receiver operation described above, the UART incorporates a special mode which provides automatic wake-up of the receiver through address frame recognition for multiprocessor communications. This mode is selected by programming bits MR1[4:3] to '11'.

In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, whose receivers are normally disabled, examine the received data stream and 'wake-up' the CPU (by setting RXRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, an address/data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1[2]: MR1[2] = 0 transmits a zero in the A/D bit position which identifies the corresponding data bits as data, while MR1[2] = 1 transmits a one in the A/D bit position which identifies the corresponding data bits as an address. The CPU should program the mode register prior to loading the corresponding data bits into the THR.

While in this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RXRDY status bit and loads the character into the RHR FIFO if the received A/D bit is a one, but discards the received character if the received A/D bit is a zero. If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SR[5]). Framing error, overrun error, and break detect operate normally whether or not the receiver is enabled.

MULTI-PURPOSE INPUT PIN

The MPI pin can be programmed as an input to one of several UART circuits. The function of the pin is selected by programming the appropriate control register (MR2[4], ACR[6:4], CSR[7:4, 3:0]). Only one of the functions may be selected at any given time. If CTS or GPI is selected, a change of state detector provided with the pin is activated. A high-to-low or low-to-high transition of the inputs lasting longer than 25 – 50μsec sets the MPI change-of-state bit in the interrupt status register. The bit is cleared via a command. The change of state can be programmed to generate an interrupt to the CPU by setting the corresponding bit in the interrupt mask register.

The input port pulse detection circuitry uses a 38.4KHz sampling clock derived from one of the baud rate generator taps. This produces a sampling period of slightly more than 25μsec (assuming a 3.6864MHz oscillator input). The detection circuitry, in order to guarantee that a true change in level has occurred, requires two successive samples at the new logic level be observed. As a consequence, the minimum duration of the signal change is 25μsec if the transition occurs coincident with the first sample pulse. The 50μsec time refers to the condition where the change of state is just missed and the first change of state is not detected until after an additional 25μsec.

MULTI-PURPOSE OUTPUT PIN

This pin can be programmed to serve as a request-to-send output, the counter/timer output, the output for the 1X or 16X transmitter or receiver clocks, the TXRDY output or the RXRDY/FFULL output (see ACR [2:0] – MPO Output Select).

REGISTERS

The operation of the UART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. Addressing of the registers is as described in table 1.

The contents of certain control registers are initialized to zero on RESET (see Reset pin description). Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems – e.g., changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. The contents of the MR, the CSR, and the ACR should only be changed while the receiver(s) and transmitter(s) are disabled, and certain changes to the ACR should only be made while the C/T is stopped.

The bit formats of the UART registers are depicted in table 2.

MR1 – Mode Register 1

MR1 is accessed when the MR pointer points to MR1. The pointer is set to MR1 by RESET or by a set pointer command applied via CR. After reading or writing MR1, the pointers are set at MR2.

MR1[7] – Receiver Request-to-Send Control

This bit controls the deactivation of the RTSN output (MPO) by the receiver. This output is manually asserted and negated by commands applied via the command register. MR1[7] = 1 causes RTSN to be automatically negated upon receipt of a valid start bit if the receiver FIFO is full. RTSN is reasserted when an empty FIFO position is available. This feature can be used to prevent overrun in the receiver by using the RTSN output signal to control the CTS input of the transmitting device.

MR1[6] – Receiver Interrupt Select

This bit selects either the receiver ready status (RXRDY) or the FIFO full status (FFULL) to be used for CPU interrupts.

MR1[5] – Error Mode Select

This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break). In the character mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the block mode, the status provided in the SR for these bits is the accumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last reset error command was issued.

MR1[4:3] – Parity Mode Select

If with parity or force parity is selected , a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1[4:3] = 11 selects the channel to operate in the special wake up mode.

MR1[2] – Parity Type Select

This bit selects the parity type (odd or even) if the with parity mode is programmed by MR1[4:3], and the polarity of the forced parity bit if the force parity mode is programmed. It has no effect if the no parity mode is programmed. In the special wake up mode, it selects the polarity of the A/D bit.

MR1[1:0] – Bits per Character Select

This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

MR2 Mode Register 2

MR2 is accessed when the channel MR pointer points to MR2, which occurs after any access to MR1. Accesses to MR2 do not change the pointer.

MR2[7:6] – Mode Select

The UART can operate in one of four modes: MR2[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2[7:6] = 01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TXD output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The TXRDY and TXEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be selected. MR2[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TXD output is held high.
4. The RXD input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2[7:6] = 11. In this mode:

1. Received data is reclocked and retransmitted on the TXD output.
2. The receive clock is used for the transmitter.
3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., the transmitted parity bit is as received.
5. The receiver must be enabled, but the transmitter need not be enabled.
6. Character framing is not checked, and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

When switching in and out of the various modes, the selected mode is activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes; if the deselection occurs just after the receiver has sampled the stop bit (indicated to be in autoecho by assertion of RXRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until one full stop bit has been retransmitted.

MR2[5] – Transmitter Request-to-Send Control

This bit controls the deactivation of the RTSN output (MPO) by the transmitter. This output

Advance Information

is manually asserted and negated by appropriate commands issued via the command register. MR2[5] = 1 causes RTSN to be reset automatically one bit time after the characters in the transmit shift register and in the THR (if any) are completely transmitted; includes the programmed number of stop bits if the transmitter is not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2[5] = 1.
2. Enable transmitter.
3. Assert RTSN via command.
4. Send message.
5. Verify the next to last character of the message is being sent by waiting until transmitter ready is asserted. Disable transmitter after the last character is loaded into the THR.
6. The last character will be transmitted and RTSN will be reset one bit time after the last stop bit.

MR2[4] – Clear-to-Send Control
The state of this bit determines if the CTSN input (MPI) controls the operation of the transmitter. If this bit is 0, CTSN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSN each time it is ready to send a character. If it is asserted (low), the character is transmitted. If it is negated (high), the TXD output remains in the marking state and the transmission is delayed until CTSN goes low. Changes in CTSN while a character is being transmitted do not affect the transmission of that character. This feature can be used to prevent overrun of a remote receiver.

MR2[3:0] – Stop Bit Length Select
This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. In all cases, the receiver only checks for a mark condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled). If an external 1X clock is used for the transmitter, MR2[3] = 0 selects one stop bit and MR2[3] = 1 selects two stop bits to be transmitted.

CSR – Clock Select Register

CSR[7:4] – Receiver Clock Select
When using a 3.6864MHz crystal or external clock input, this field selects the baud rate clock for the receiver as shown in table 3.

Table 2. REGISTER BIT FORMATS

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RX RTS CONTROL	RX INT SELECT	ERROR MODE	PARITY MODE		PARITY TYPE	BITS PER CHAR	
MR1	0=no 1=yes	0=RXRDY 1=FFULL	0=char 1=block	00=with parity 01=force parity 10=no parity 11=special mode		0=even 1=odd	00=5 01=6 10=7 11=8	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	CHANNEL MODE		Tx RTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
MR2	00=Normal 01=Auto echo 10=Local loop 11=Remote loop		0=no 1=yes	0=no 1=yes	0=0.563 1=0.625 2=0.688 3=0.750	4=0.813 5=0.875 6=0.938 7=1.000	8=1.563 9=1.625 A=1.688 B=1.750	C=1.813 D=1.875 E=1.938 F=2.000

*Add 0.5 to values shown for 0 – 7, if channel is programmed for 5 bits/char.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
CSR	See Text				See Text			

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	MISCELLANEOUS COMMANDS				DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
CR	See Text				0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

Advance Information

Table 2. REGISTER BIT FORMATS (Continued)

SR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
RECEIVED BREAK	FRAMING ERROR	PARITY ERROR	OVERRUN ERROR	TXEMT	TXRDY	FFULL	RXRDY
0=no 1=yes *	0=no 1=yes *	0=no 1=yes *	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

*These status bits are appended to the corresponding data character in the receive FIFO. A read of the status register provides these bits [7:5] from the top of the FIFO together with bits [4:0]. These bits are cleared by a reset error status command. In character mode they are reset when the corresponding data character is read from the FIFO.

ACR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			POWER DOWN MODE	MPO PIN FUNCTION SELECT		
0=set1 1=set2	See Text			0 = on 1 = off	000=RTSN 001=C/TO 010=TXC(1X) 011=TXC(16X)	100=RXC(1X)) 101=RXC(16X) 110=TXRDY 111=RXRDY/FFULL	

ISR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
MPI PIN CHANGE	MPI PIN CURRENT STATE		COUNTER READY	DELTA BREAK	RXRDY/ FFULL	TXEMT	TXRDY
0=no 1=yes	0=low 1=high	not used	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes	0=no 1=yes

IMR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
MPI CHANGE INT	MPI LEVEL INT		COUNTER READY! nIINT	DELTA BREAK INT	RXRDY/ FFULL INT	TXEMT INT	TXRDY! nIINT
0=off 1=on	0=off 1=on	not used	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on	0=off 1=on

CTUR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]

CTLR

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]

Advance Information

Table 3. BAUD RATE

CSR[7:4]	ACR[7]=0	ACR[7]=1
0 0 0 0	50	75
0 0 0 1	110	110
0 0 1 0	134.5	134.5
0 0 1 1	200	150
0 1 0 0	300	300
0 1 0 1	600	600
0 1 1 0	1,200	1,200
0 1 1 1	1,050	2,000
1 0 0 0	2,400	2,400
1 0 0 1	4,800	4,800
1 0 1 0	7,200	1,800
1 0 1 1	9,600	9,600
1 1 0 0	38.4K	19.2K
1 1 0 1	Timer	Timer
1 1 1 0	MPI – 16X	MPI – 16X
1 1 1 1	MPI – 1X	MPI – 1X

The receiver clock is always a 16X clock, except for CSR[7:4]=1111.

CSR[3:0] – Transmitter Clock Select
This field selects the baud rate clock for the transmitter. The field definition is as shown in table 3.

CR – Command Register

CR is used to write commands to the UART. Multiple commands can be specified in a single write to CR as long as the commands are non-conflicting, e.g., the enable transmitter and reset transmitter commands cannot be specified in a single command word.

CR[7:4] – Miscellaneous Commands
The encoded value of this field may be used to specify a single command as follows:

0000 No command.

0001 Reset MR pointer. Causes the MR pointer to point to MR1.

0010 Reset receiver. Resets the receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.

0011 Reset transmitter. Resets the transmitter as if a hardware reset had been applied.

0100 Reset error status. Clears the received break, parity error, framing error, and overrun error bits in the status register (SR[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared), and in block mode to clear all error status after a block of data has been received.

0101 Reset break change interrupt. Causes the break detect change bit in the interrupt status register (ISR[3]) to be cleared to zero.

0110 Start break. Forces the TXD output low (spacing). If the transmitter is empty, the start of the break condition will be delayed up to two bit times. If the transmitter is active, the break begins when transmission of the character is completed. If a character is in the THR, the start of break is delayed until that character or any others loaded after it has been transmitted (TXEMT must be true before break begins). The transmitter must be enabled to start a break.

0111 Stop break. The TXD line will go high (marking) within two bit times. TXD will remain high for one bit time before the next character, if any, is transmitted.

1000 Start C/T. In counter or timer modes, causes the contents of CTUR/ CTLR to be preset into the counter/timer and starts the counting cycle. In timer mode, any counting cycle in progress when the command is issued is terminated. In counter mode, has no effect unless a stop C/T command was issued previously.

1001 Stop counter. In counter mode, stops operation of the counter/timer, resets the counter ready bit in the ISR, and forces the MPO output high if it is programmed to be the output of the C/T. In timer mode, resets the counter ready bit in the ISR but has no affect on the counter/timer itself or on the MPO output.

1010 Assert RTSN. Causes the RTSN output to be asserted (low).

1011 Negate RTSN. Causes the RTSN output to be negated (high).

1100 Reset MPI change interrupt. Causes the MPI change bit in the interrupt status register (ISR[7]) to be cleared to zero.

1101 Reserved.

111x Reserved.

CR[3] – Disable Transmitter
This command terminates transmitter operation and resets the TXRDY and TXEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CR[2] – Enable Transmitter
Enables operation of the channel A transmitter. The TXRDY status bit will be asserted.

CR[1] – Disable Receiver
This command terminates operation of the receiver immediately – a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special wakeup mode is programmed, the receiver operates even if it is disabled (see Wakeup Mode).

CR[0] – Enable Receiver
Enables operation of the receiver. If not in the special wakeup mode, this also forces the receiver into the search for start bit state.

SR – Channel Status Register

SR[7] – Received Break
This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received; further entries to the FIFO are inhibited until the RXD line returns to the marking state for at least one half bit time (two successive edges of the internal or external 1x clock).

When this bit is set, the change in break bit in the ISR (ISR[3]) is set. ISR[3] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry is capable of detecting breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must last until the end of the next character time in order for it to be detected.

SR[6] – Framing Error (FE)
This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SR[5] – Parity Error (PE)
This bit is set when the with parity or force parity mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special wakeup mode, the parity error bit stores the received A/D bit.

SR[4] – Overrun Error (OE)
This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a reset error status command.

SR[3] – Transmitter Empty (TXEMT)
This bit will be set when the transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. However, this bit is not set until one character has been transmitted. It is set after transmission of the last stop bit of a

character, if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU, or when the transmitter is disabled.

SR[2] – Transmitter Ready (TXRDY)
This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TXRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled, e.g., characters loaded in the THR while the transmitter is disabled will not be transmitted.

SR[1] – FIFO Full (FFULL)
This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the FIFO and there is no character in the receive shift register. If a character is waiting in the receive shift register because the FIFO is full, FFULL is not reset after reading the FIFO once.

SR[0] – Receiver Ready (RXRDY)
This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR, and no more characters are in the FIFO.

ACR – Auxiliary Control Register

ACR[7] – Baud Rate Generator Set Select
This bit selects one of two sets of baud rates generated by the BRG.

Set 1: 50, 110, 134.5, 200, 300, 600, 1.05K, 1.2K, 2.4K, 4.8K, 7.2K, 9.6K, and 38.4K baud.

Set 2: 75, 110, 134.5, 150, 300, 600, 1.2K, 1.8K, 2.0K, 2.4K, 4.8K, 9.6K, and 19.2K baud.

The selected set of rates is available for use by the receiver and transmitter.

ACR[6:4] – Counter/Timer Mode and Clock Source Select
This field selects the operating mode of the counter/timer and its clock source as follows:

ACR[6:4]	Mode	Clock Source
0 0 0	Counter	MPI pin
0 0 1	Counter	MPI pin divided by 16
0 1 0	Counter	TXC – 1X clock of the transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	MPI pin
1 0 1	Timer	MPI pin divided by 16
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

ACR[3] – Power Down Mode Select
This bit, when set to zero, selects the power down mode. In this mode, the 2691 oscillator is stopped and all functions requiring this clock are suspended. The contents of all registers are saved. It is recommended that the transmitter and receiver be disabled prior to placing the 2691 in this mode. Note that this bit must be set to a logic 1 after power up.

ACR[2:0] – MPO Output Select
This field programs the MPO output pin to provide one of the following:

000 Request to send active low output (RTSN). This output is asserted and negated via the command register. Mode RTSN can be programmed to be automatically reset after the character in the transmitter is completely shifted out or when the receiver FIFO and receiver shift register are full using MR2[5] and MR1[7], respectively.

001 The counter/timer output. In the timer mode, this output is a square wave with a period of twice the value (in clock periods) of the contents of the CTUR and CTLR. In the counter mode, the output remains high until the terminal count is reached, at which time it goes low. The output returns to the high state when the counter is stopped by a stop counter command.

010 The 1X clock for the transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a non-synchronized 1X clock is output.

011 The 16X clock for the transmitter. This is the clock selected by CSR[3:0], and is a 1X clock if CSR[3:0] = 1111.

100 The 1X clock for the receiver, which is the clock that samples the received data. If data is not being received, a non-synchronized 1X clock is output.

101 The 16X clock for the receiver. This is the clock selected by CSR[7:4], and is a 1X clock if CSR[7:4] = 1111.

110 The transmitter register empty signal, which is the same as SR[2].

111 The receiver ready or FIFO full signal (complement of ISR[2]).

ISR – Interrupt Status Register

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output is asserted (low). If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no affect on the INTRN output. Note that the IMR does not mask the reading of the ISR – the true status is provided regardless of the contents of the IMR.

ISR[7] – MPI Change of State
This bit is set when a change of state occurs at the MPI input pin. It is reset by a reset MPI change interrupt command.

ISR[6] – MPI Current State
This bit provides the current state of the MPI pin. The information is unlatched and reflects the state of the pin at the time the ISR is read.

ISR[4] – Counter Ready
In the counter mode of operation, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command. It is initialized to '0' when the chip is reset.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time the C/T reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the C/T.

ISR[3] – Change in Break
This bit, when set, indicates that the receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a reset break change interrupt command.

ISR[2] – Receiver Ready or FIFO Full
The function of this bit is programmed by MR1[6]. If programmed as receiver ready, it indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the receiver FIFO. If the FIFO contains more characters, the bit will be set again after the FIFO is read. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when FIFO is read and there is no character in the receiver shift register. If there is a character waiting in the receive shift register because the FIFO is full, the bit is set

Advance Information

again when the waiting character is transferred into the FIFO.

ISR[1] – Transmitter Empty
This bit is a duplicate of TXEMT (SR[3]).

ISR[0] – Transmitter Ready
This bit is a duplicate of TXRDY (SR[2]).

IMR – Interrupt Mask Register

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is a '1', the INTRN output is asserted (low). If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask reading of the ISR.

CTUR AND CTLR – Counter/Timer Registers

The CTUR and CTLR hold the eight MSB's and eight LSB's, respectively, the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded is 0002_{16}.

In the timer (programmable divider) mode, the C/T generates a square wave whose period is twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half-periods will be.

The counter ready status bit (ISR[4]) is set once each cycle of the square wave. The bit is reset by a stop counter command. The command, however, does not stop the C/T. The generated square wave is output on MPO if it is programmed to be the C/T output.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR. Counting begins upon receipt of a start C/T command. Upon reaching the terminal count, the counter ready interrupt bit (ISR[4]) is set. The counter continues counting past the terminal count until stopped by the CPU. If MPO is programmed to be the output of the C/T, the output remains high until the terminal count is reached, at which time it goes low.

The output returns to the high state and ISR[4] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

Advance Information

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
Voltage from V_{CC} to GND[3]	−0.5 to +7.0	V
Voltage from any pin to ground[3]	−0.5 to V_{CC} ±10%	V
Power dissipation	2.0	W

DC ELECTRICAL CHARACTERISTICS T_A=0°C to +70°C, V_{CC}=5.0V ±10%[4,5,6]

	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage					
	All except X1/CLK		2.0			V
	X1/CLK		0.9V_{CC}		V_{CC}	V
V_{OL}	Output low voltage	I_{OL} = 2.4mA			0.4	V
V_{OH}[7]	Output high voltage					
	(except open drain outputs)	I_{OH} = −400μA	2.4			V
I_{IL}	Input leakage current	V_{IN} = 0 to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	V_O = 0 to V_{CC}	−10		10	μA
I_{OD}	Open drain output leakage current	V_O = 0 to V_{CC}			10	μA
I_{X1L}	X1/CLK low input current	V_{IN} = 0, X2 floated	−100	−30	0.0	μA
I_{X1H}	X1/CLK high input current	V_{IN} = V_{CC}, X2 floated	0.0	+30	100	μA
I_{X2L}	X2 low input current	V_{IN} = 0, X1 floated	−10.0	−3.5	0.0	mA
I_{X2H}	X2 high input current	V_{IN} = V_{CC}, X1 floated	0.0	3.5	10	mA
I_{CC}	Power supply current				20	mA
	Standby				500	μA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature and thermal resistance of 60°C/W junction to ambient for ceramic package (116°C/W for plastic package).
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. For X1/CLK, this swing is between 0.4V and 4.4V. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at + 25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: C_L=150pF, except interrupt outputs. Test conditions for interrupt outputs: C_L=50pF, R_L=2.7Kohm to V_{CC}.
8. Timing is illustrated and referenced to the WRN and RDN inputs. The device may also be operated with CEN as the 'strobing' input. In this case, all timing specifications apply referenced to the falling and rising edges of CEN. CEN and RDN (also CEN and WRN) are OR'ed internally. As a consequence, the signal asserted last initiates the cycle and the signal negated first terminates the cycle.
9. If CEN is used as the 'strobing' input, this parameter defines the minimum high time between one CEN and the next. The RDN signal must be negated for t_{RWD} to guarantee that any status register changes are valid.
10. Consecutive write operations to the same command require at least three edges of the X1 clock between writes.

Advance Information

AC ELECTRICAL CHARACTERISTICS T_A=0°C to +70°C, V_{CC}=5.0V ±10%[4,5,6,7]

PARAMETER		TENTATIVE LIMITS			UNIT
		Min	Typ	Max	
Reset timing (figure 1)					
t_{RES}	RESET pulse width	1.0			μs
Bus timing (figure 2)[8]					
t_{AS}	A0 – A2 set-up time to RDN, WRN low	10			ns
t_{AH}	A0 – A2 hold time from RDN, WRN high	0			ns
t_{CS}	CEN set-up time to RDN, WRN low	0			ns
t_{CH}	CEN hold time from RDN, WRN high	0			ns
t_{RW}	WRN, RDN pulse width	225			ns
t_{DD}	Data valid after RDN low			175	ns
t_{DF}	Data bus floating after RDN high			100	ns
t_{DS}	Data set-up time before WRN high	100			ns
t_{DH}	Data hold time after WRN high	10			ns
t_{RWD}[10]	Time between READs and/or WRITEs	200			ns
MPI and MPO timing (figure 3)[8]					
t_{PS}	MPI input set-up time before RDN low	0			ns
t_{PH}	MPI input hold time after RDN high	0			ns
t_{PD}	MPO output valid after WRN high			370	ns
Interrupt timing (figure 4)					
t_{IR}	INTRN high from:				
	Read RHR (RXRDY/FFULL interrupt)			370	ns
	Write THR (TXRDY, TXEMT interrupt)			370	ns
	Reset command (Break change interrupt)			370	ns
	Reset command (MPI change interrupt)			370	ns
	Stop C/T command (counter interrupt)			370	ns
	Write IMR (clear of interrupt mask bit)			270	ns
Clock timing (figure 5)					
t_{CLK}	X1/CLK high or low time	100			ns
f_{CLK}	X1/CLK frequency	2.0	3.6864	4.0	MHz
t_{CTC}	CTCLK high or low time	100			ns
f_{CTC}	CTCLK frequency	0		4.0	MHz
t_{RX}	RXC high or low time	220			ns
f_{RX}	RXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
t_{TX}	TXC high or low time	220			ns
f_{TX}	TXC frequency (16X)	0		2.0	MHz
	(1X)	0		1.0	MHz
Transmitter timing (figure 6)					
t_{TXD}	TXD output delay from TxC low			350	ns
t_{TCS}	TXC output delay from TxD output data	0		150	ns
Receiver timing (figure 7)					
t_{RXS}	RXD data set-up time to RXC high	240			ns
t_{RXH}	RXD data hold time from RXC high	200			ns

Advance Information

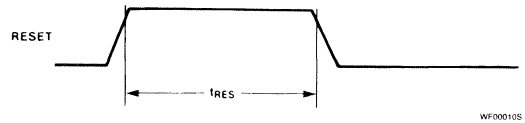


Figure 1. Reset Timing

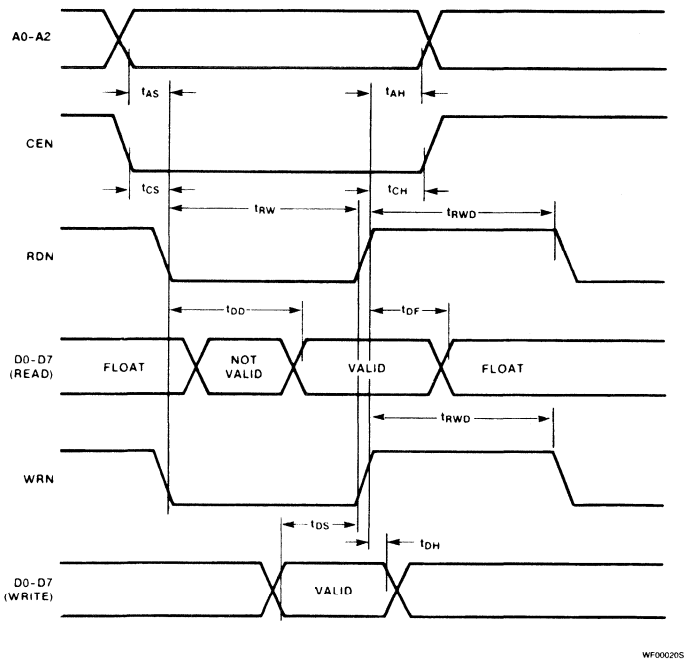


Figure 2. Bus Timing

Advance Information

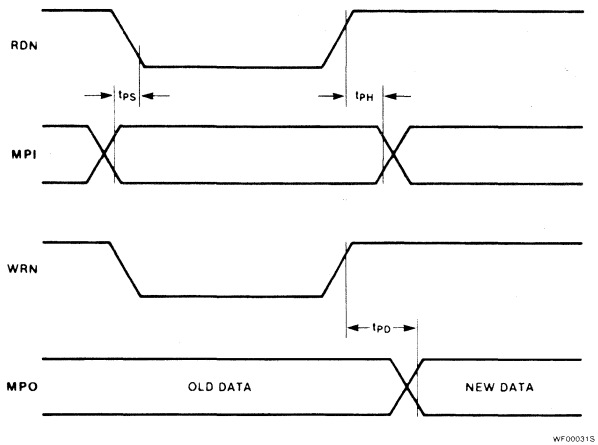


Figure 3. I/O Timing

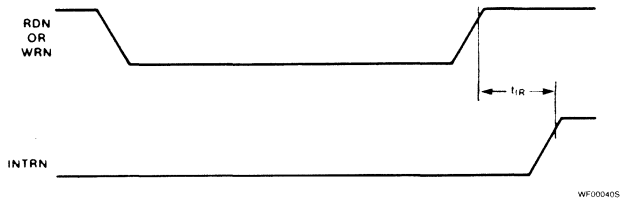


Figure 4. Interrupt Timing

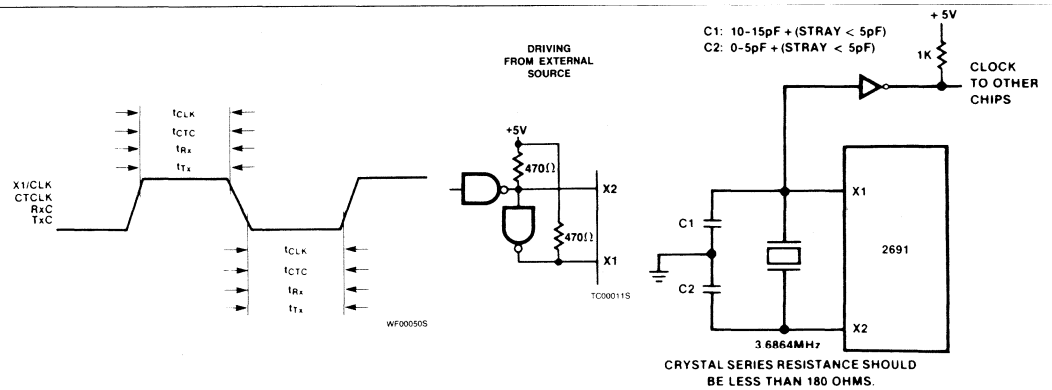


Figure 5. Clock Timing

Advance Information

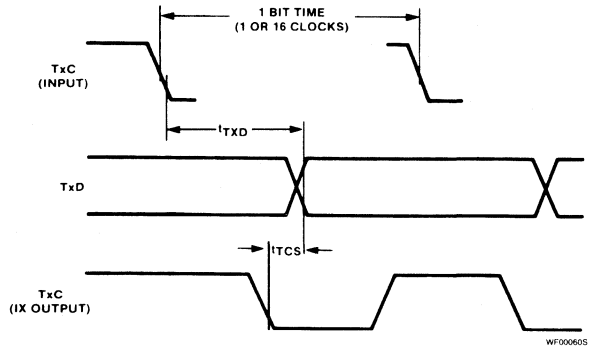


Figure 6. Transmit Timing

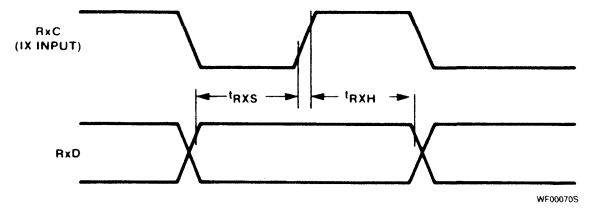
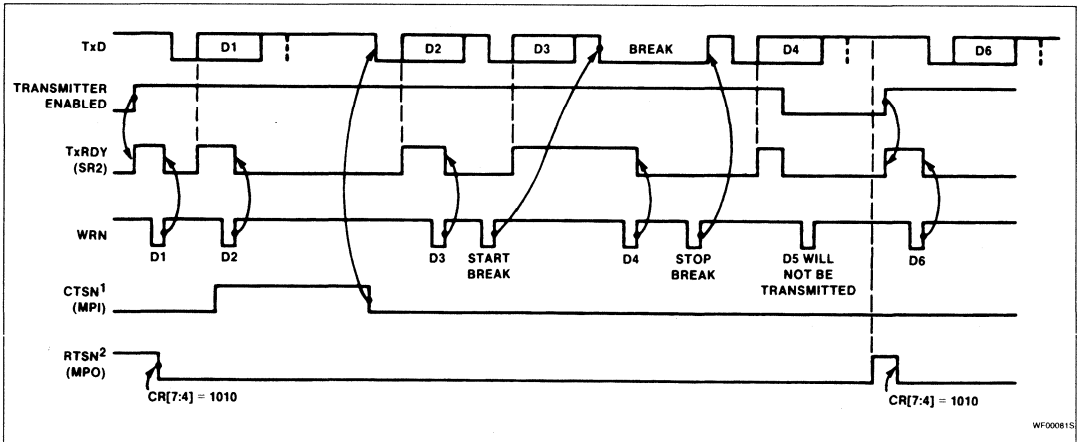


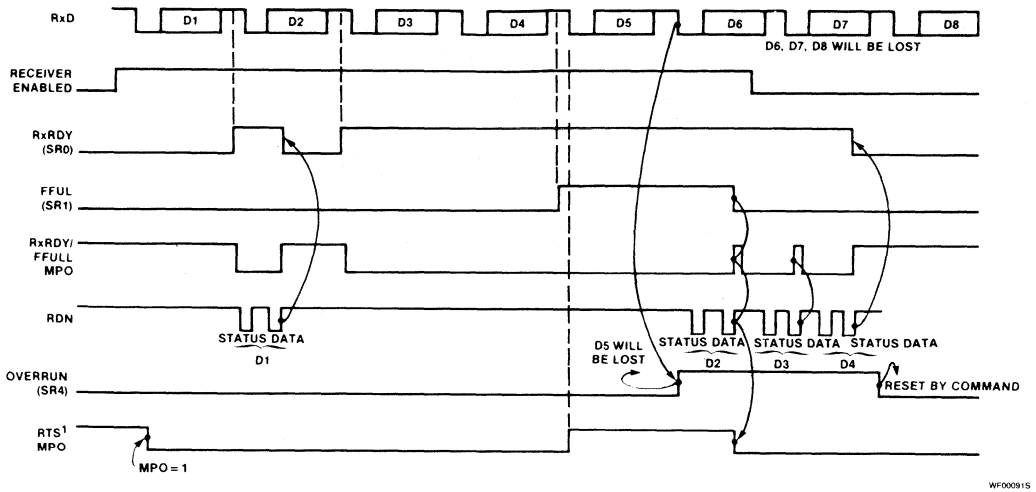
Figure 7. Receive Timing



NOTES:
1. Timing shown for MR2[4] = 1.
2. Timing shown for MR2[5] = 1.

Figure 8. Transmitter Timing

Advance Information



NOTES:
1. Timing shown for MR1[7] = 1.
2. Shown for ACR[2:0] = 111 and MR1[6] = 0.

Figure 9. Receiver Timing

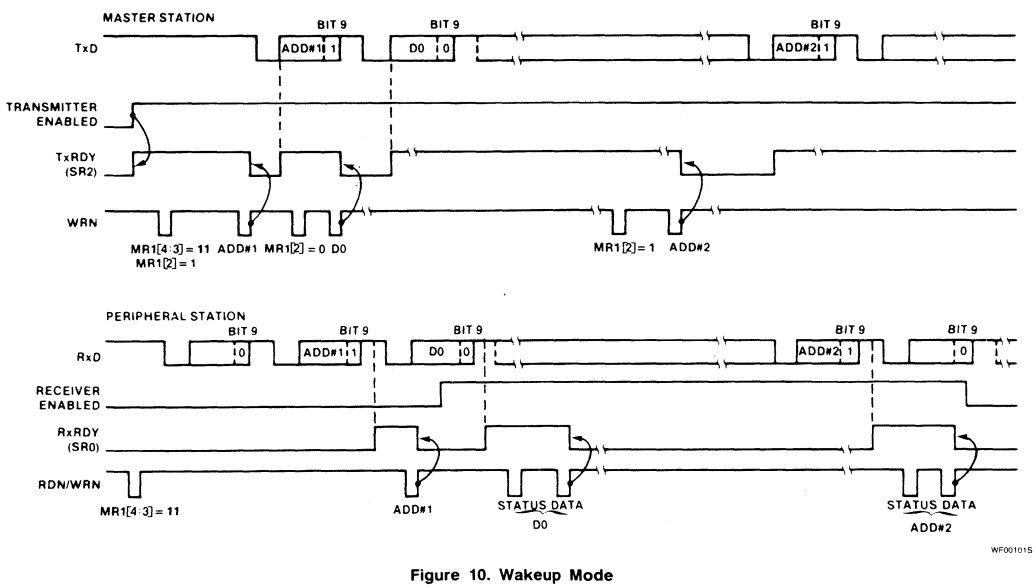


Figure 10. Wakeup Mode

Asynchronous Communications Interface

Originally published by Signetics February 1985

Product Specification

DESCRIPTION

The Signetics SCN2641 is a universal asynchronous data communications controller chip that interfaces directly to most 8-bit microprocessors and may be used in a polled or interrupt-driven system environment. The SCN2641 accepts programmed instructions from the microprocessor while supporting asynchronous serial data communications in full- or half-duplex mode.

The SCN2641 serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The SCN2641 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode.

The SCN2641 is constructed using Signetics n-channel silicon gate depletion load technology and is packaged in a 24-pin DIP.

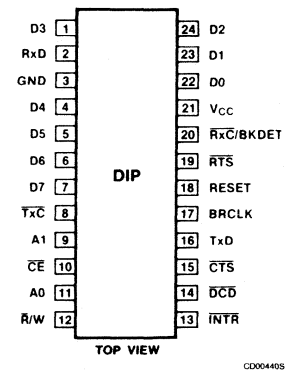
FEATURES

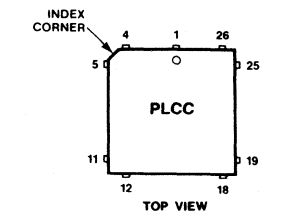
- **5- to 8-bit characters plus parity**
- **1, 1½ or 2 stop bits transmitted**
- **Odd, even or no parity**
- **Parity, overrun and framing error detection**
- **Line break detection and generation**
- **False start bit detection**
- **Automatic serial echo mode (echoplex)**
- **Local or remote maintenance loopback mode**
- **Baud rate:**
  - **DC to 1M bps (1X clock)**
  - **DC to 62.5K bps (16X clock)**
  - **DC to 15.625K bps (64X clock)**
- **Internal or external baud rate clock**
- **16 internal rates**
- **Double-buffered transmitter and receiver**
- **Single +5V power supply**
- **400 mil package width**

APPLICATIONS

- **Intelligent terminals**
- **Network processors**
- **Front-end processors**
- **Remote data concentrators**
- **Serial peripherals**

PIN CONFIGURATION



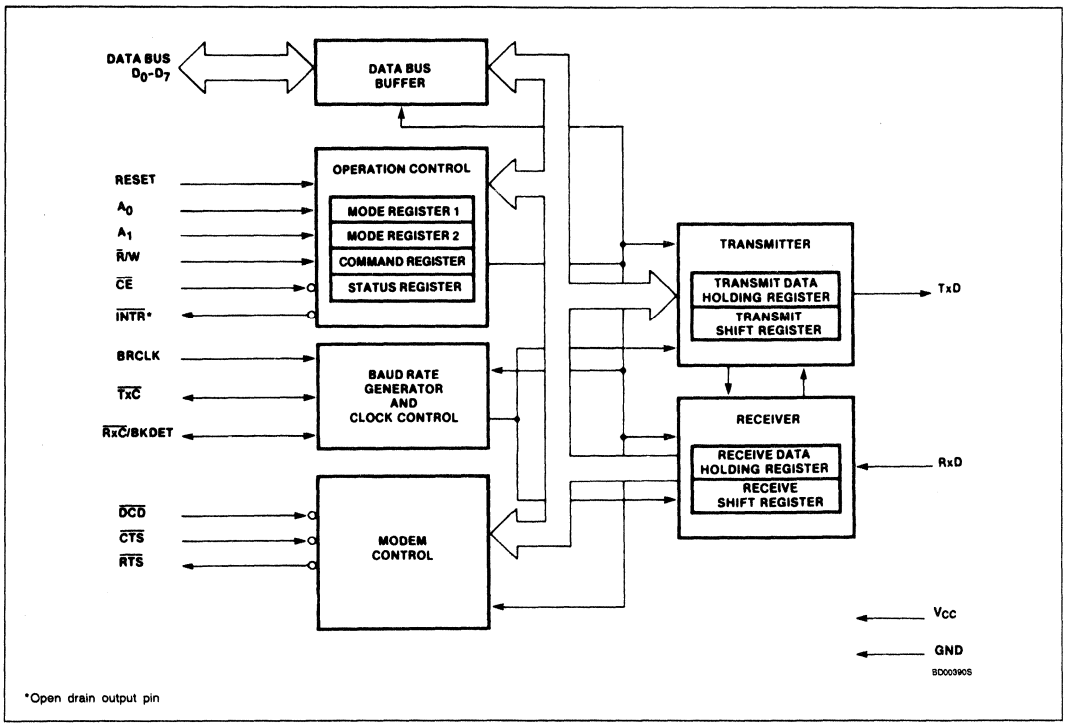


Pin	Function	Pin	Function
1	D2	15	INTR
2	D3	16	DCD
3	RxD	17	CTS
4	GND	18	NC
5	D4	19	TxD
6	D5	20	BRCLK
7	D6	21	RESET
8	NC	22	NC
9	D7	23	RTS
10	TxC	24	NC
11	A1	25	RxC/BKDET
12	CE	26	V_{CC}
13	A0	27	D0
14	R/W	28	D1

Product Specification

ORDERING CODE

PACKAGES	V_{CC} = 5V±5%, T_A = 0°C to 70°C
Plastic DIP	SCN2641CC1N24
Plastic LCC	SCN2641CC1A28



*Open drain output pin

BLOCK DIAGRAM

The SCN2641 consists of five major sections. These are the transmitter, receiver, timing, operation control and modem control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers 1 and 2, the command register, and the status register. Details of register addressing are presented in the SCN2641 programming section of this data sheet.

Timing

The SCN2641 contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full-duplex operation. See table 1.

Receiver

The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for certain errors and sends an "assembled" character to the CPU.

Transmitter

The transmitter accepts parallel data from the CPU, appends start and stop bits, and, optionally, a parity bit, and outputs a composite serial stream of data on the TxD output pin.

Modem Control

The modem control section provides interfacing for two input signals and one output signal used for "handshaking" and status indication between the CPU and a modem.

INTERFACE SIGNALS

The SCN2641 interface signals can be grouped into two types: the CPU-related signals (shown in table 2), which interface the SCN2641 to the microprocessor system and the device-related signals (shown in table 3), which are used to interface to the communications device or system.

OPERATION

The functional operation of the SCN2641 is programmed by a set of control words supplied by the CPU. These control words speci-

Product Specification

Table 1. BAUD RATE GENERATOR CHARACTERISTICS (BRCLK = 3.6864MHz)

MR23 – 20	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8kHz	—	4608
0001	75	1.2	—	3072
0010	110	1.7596	-0.022	2095
0011	134.5	2.152	—	1713
0100	150	2.4	—	1536
0101	300	4.8	—	768
0110	600	9.6	—	384
0111	1200	19.2	—	192
1000	1800	28.8	—	128
1001	2000	32.055	0.174	115
1010	2400	38.4	—	96
1011	3600	57.6	—	64
1100	4800	76.8	—	48
1101	7200	115.2	—	32
1110	9600	153.6	—	24
1111	19200	307.2	—	12

fy items such as baud rate, number of bits per character, etc. The programming procedure is described in the SCN2641 programming section of this data sheet.

After programming, the SCN2641 is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The SCN2641 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. The receiver looks for a high-to-low transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and one stop bit have been assembled. The data is then transferred to the receive data holding register, the RxRDY bit in the status register is set, and the $\overline{INTR}$ output is asserted. If the character length is less than 8 bits, the high-order unused bits in the holding register are set to zero. The parity error, framing error, and overrun error status bits are strobed into the status register on the positive-going edge of $\overline{RxC}$ corresponding to the received character boundary. If the stop bit is present, the receiver will immediately begin its search for the next start bit. If the stop bit is absent (framing error), the receiver will interpret a space as a start bit if it persists into the next bit time interval. If a break condition is detected (RxD is low for the entire character as well as the stop bit), only one character consisting of all zeros (with the FE status bit set) will be transferred to the holding register. The RxD input must return to a high condition before a search for the next start bit begins.

Pin 20 can be programmed to be a break detect output by appropriate setting of MR27 – MR24. If so, a detected break will cause that pin to go high. When RxD returns to mark for one RxC time, pin 20 will go low. Refer to the break detection timing diagram.

Transmitter

The SCN2641 is conditioned to transmit data when the $\overline{CTS}$ input is low and the TxEN command register bit is set. The SCN2641 indicates to the CPU that it can accept a character for transmission by setting the TxRDY status bit and asserting the $\overline{INTR}$ output. When the CPU writes a character into the transmit data holding register, these conditions are negated. Data is transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

The transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the transmit holding register, the TxD output remains in the marking (high) condition and the TxEMT/DSCHG status bit and the $\overline{INTR}$ output are asserted. Transmission resumes when the CPU loads a new character into the holding register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the send break command bit (CR3) high.

Product Specification

Table 2. CPU-RELATED SIGNALS

PIN NAME	PIN NO. DIP	PIN NO. PLCC	INPUT/ OUTPUT	FUNCTION
V_{CC}	21	26	I	+5V supply input
GND	3	4	I	Ground
RESET	18	21	I	A high on this input performs a master reset on the SCN2641. This signal asynchronously terminates any device activity and clears the mode, command and status registers. The device assumes the idle state and remains there until initialized with appropriate control words.
$A_1 - A_0$	9, 11	11, 13	I	Address lines used to select internal SCN2641 registers.
$\overline{R}/W$	12	14	I	Read command when low, write command when high.
$\overline{CE}$	10	12	I	Chip enable command. When low, indicates that control and data lines to the SCN2641 are valid and that the operation specified by the $\overline{R}/W$, A_1 and A_0 inputs should be performed. When high, places the $D_0 - D_7$ lines in the three-state condition.
$D_7 - D_0$	7-4,1, 24-22	9,7-5, 2,1, 28,27	I/O	8-bit, three-state data bus used to transfor commands, data and status between the SCN2641 and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{INTR}$	13	15	O	Interrupt request output (open drain). This output is asserted (low) under the following conditions. 1. When the transmitter holding register (THR) is ready to accept a data character from the CPU. This corresponds to assertion of status bit SR0. If this is the only condition asserting the output, the output will be negated (high) when the THR is loaded by the CPU, or if the transmitter is disabled via command register bit CR0. 2. When the receiver holding register (RHR) has a character ready to be read by the CPU. This corresponds to assertion of status bit SR1. If this is the only condition asserting the output, the output will be negated (high) when the RHR is read by the CPU, or if the receiver is disabled via command register bit CR2. 3. When the transmitter has completed serialization of the last character loaded by the CPU. This corresponds to assertion of status bit SR2. If this is the only condition asserting the output, the output will be negated (high) when the THR is loaded by the CPU. 4. When a change of state has occurred at the $\overline{DCD}$ input while either the receiver or the transmitter are enabled. This corresponds to assertion of status bit SR2. If this is the only condition asserting the output, the output will be negated (high) when the status register is read by the CPU.

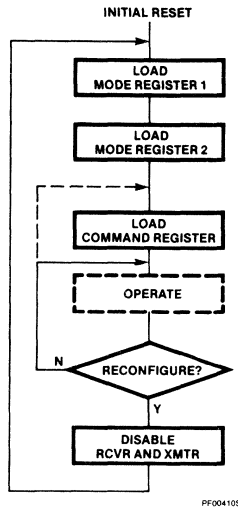
PROGRAMMING

Prior to initiating data communications, the SCN2641 operational mode must be programmed by performing write operations to the mode and command registers. The SCN2641 can be reconfigured at any time during program execution. A flowchart of the initialization process appears in figure 1.

The internal registers of the SCN2641 are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in table 4.

Reading or loading the mode registers is done as follows: the first write (or read) operation addresses mode register 1 and a subsequent operation addresses mode register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointer is reset to mode register 1 by a RESET input or by performing a read command register operation, but is unaffected by any other read or write operation.

The SCN2641 register formats are summarized in tables 5, 6, 7 and 8. Mode registers 1 and 2 define the general operational characteristics of the SCN2641, while the command register controls the operation within this basic framework. The SCN2641 indicates its status in the status register. These registers are cleared when a RESET input is applied.



NOTE:
Mode Register 1 must be written before 2 can be written. Mode register 2 need not be programmed if external clocks are used.

Figure 1. SCN2641 Initialization Flowchart

Mode Register 1 (MR1)

Table 5 illustrates mode register 1. Bits MR11 and MR10 select the baud rate multiplier. 1X, 16X and 64X multipliers are programmable if the external clock input option is selected by MR24 or MR25.

MR13 and MR12 select a character length of 5, 6, 7 or 8 bits. The character length does not include the parity bit, if programmed, and does not include the start and stop bits.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

MR17 and MR16 select character framing of 1, 1.5 or 2 stop bits. (If 1X baud rate is

programmed, 1.5 stop bits default to 1 stop bit on transmit.)

The bits in the mode register affecting character assembly and disassembly (MR12 – MR16) can be changed dynamically (during active receive/transmit operation). The character mode register affects both the transmitter and receiver; therefore, character changes should be made when RxEN and TxEN = 0 or when TxEN = 1 and the transmitter is marking in half-duplex mode (RxEN = 0).

To effect assembly/disassembly of the next received/transmitted character, MR12 – MR15 must be changed within n-bit times of the assertion of RxRDY/TxRDY. (n = smaller of the new and old character lengths.)

Mode Register 2 (MR2)

Table 6 illustrates mode register 2. MR23, MR22, MR21 and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable as per table 1. MR23 – MR20 are don't cares if external clocks are selected (MR25 – MR24 = 0). The individual rates are given in table 1.

MR24 – MR27 select the receive and transmit clock source (either the BRG or an external input) and the function at pins 8 and 20. Refer to table 6.

Command Register (CR)

Table 7 illustrates the command register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. If the transmitter is disabled, it will complete the transmission of the character in the transmit shift register (if any) prior to terminating operation. The TxD output will then remain in the marking state (high), while the TxRDY and TxEMT status bits go low. Disabling the receiver causes the RxRDY status bit to go low. If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be negated. A 0-to-1 transition of CR2 will initiate start bit search on the second $\overline{RxC}$ rising edge following the transition.

Bit CR5 (RTS) controls the RTS output. Data at the output is the logical complement of the register data.

Setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The user should wait at least one bit time after terminating the break before loading the THR with the next character to be transmitted.

Setting CR4 causes the error flags in the status register (SR3, SR4 and SR5) to be cleared. This is a one-time command. There is no internal latch for this bit.

Table 3. DEVICE-RELATED SIGNALS

PIN NAME	DIP	PLCC	INPUT/ OUTPUT	FUNCTION
BRCLK	17	20	I	Clock input to the internal baud rate generator (see table 1). Not required if external receiver and transmitter clocks are used.
$\overline{RxC}$/ BKDET	20	25	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. Data are sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin can be a 1X/16X clock or a break detect output pin.
RxD	2	3	I	Serial data input to the receiver. "Mark" is high, "space" is low.
$\overline{TxC}$	8	10	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by mode register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, this pin can be a 1X/16X clock output.
TxD	16	19	O	Serial data output from the transmitter. "Mark" is high, "space" is low. Held in mark condition when the transmitter is disabled.
$\overline{DCD}$	14	16	I	Data carrier detect input. Must be low in order for the receiver to operate. Its complement appears as status register bit SR6. Causes a low output on $\overline{INTR}$ when its state changes if CR2 or CR0 = 1. If DCD goes high while receiving, the RxC is internally inhibited. Operation of the receiver resumes on the second $\overline{RxC}$ rising edge following assertion of DCD.
$\overline{CTS}$	15	17	I	Clear to send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the transmit shift register will be transmitted before termination.
$\overline{RTS}$	19	23	O	General-purpose output which is the complement of command register bit CR5. Normally used to indicate request to send. See Command Register (CR5) for details.

Table 4. REGISTER ADDRESSING

$\overline{CE}$	A_1	A_0	$\overline{R}/W$	FUNCTION
1	X	X	X	Three-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Invalid
0	1	0	0	Read mode registers ½
0	1	0	1	Write mode registers ½
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE:
See AC characteristics section for timing requirements.

When CR5 (RTS) is set, the RTS pin is forced low. A 1-to-0 transition of CR5 will cause $\overline{RTS}$ to go high (inactive) one TxC time after the last serial bit has been transmitted. If a 1-to-0 transition of CR5 occurs while data is being transmitted, $\overline{RTS}$ will remain low (active) until both the THR and the transmit shift register are empty and then go high one TxC time later.

The SCN2641 can operate in one of four submodes. The operational submode is determined by CR7 and CR6. CR7 – CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the mode and status register instructions.

CR7 – CR6 = 01 places the SCN2641 in the automatic echo mode. Clocked, regenerated received data are automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU-to-receiver communications continue normally, but the CPU-to-transmitter link is disabled. Only the first character of a break condition is echoed. The TxD output will go high until the next valid start is detected. The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. The $\overline{INTR}$ pin will reflect only the data set change condition.
4. The TxEN command (CR0) is ignored.

Two diagnostic submodes can also be configured. In local loopback mode (CR7 – CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{RTS}$ is connected to $\overline{CTS}$.
3. The receiver is clocked by the transmitter clock.
4. The $\overline{RTS}$ and TxD outputs are held high.
5. The $\overline{CTS}$, $\overline{DCD}$ and RxD inputs are ignored.

Additional requirements to operate in the local loopback mode are that CR0 (TxEN), CR1 and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the SCN2641.

The second diagnostic mode is the remote loopback mode (CR7 – CR6 = 11). In this mode:

1. Data assembled by the receiver are automatically placed in the transmit holding register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data is sent to the local CPU, but the error status conditions (PE, OE, FE) are set.
4. The $\overline{INTR}$ output is held high.
5. CR0 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the status register (as shown in table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the transmitter ready (TxRDY) status bit. It is valid only when the transmitter is enabled. If equal to 0, it indicates that the transmit holding register has been loaded by the CPU and the data has not been transferred to the transmit shift register. If set equal to 1, it indicates that the holding register is ready to accept data from the CPU. This bit is initially set when the transmitter is enabled by CR0, unless a character has previously been loaded into the holding register. It is not set when the automatic echo or remote loopback modes are programmed. When this bit is set, the $\overline{INTR}$ output pin is low, except in the automatic echo and remote loopback modes.

SR1, the receiver ready (RxRDY) status bit, indicates the condition of the receive data holding register. If set, it indicates that a character has been loaded into the holding register from the receive shift register and is ready to be read by the CPU. If equal to 0, there is no new character in the holding register. This bit is cleared when the CPU reads the receive data holding register or when the receiver is disabled by CR2. When set, the $\overline{INTR}$ output is low.

Table 5. MODE REGISTER 1 (MR1)

MR17	MR16	MR15	MR14	MR13	MR12	MR11	MR10
Stop Bit Length		Parity Type	Parity Control	Character Length		Mode and Baud Rate Factor	
00 = Invalid 01 = 1 stop bit 10 = 1 1/2 stop bits 11 = 2 stop bits		0 = Odd 1 = Even	0 = Disabled 1 = Enabled	00 = 5 bits 01 = 6 bits 10 = 7 bits 11 = 8 bits		00 = Invalid 01 = 1X rate 10 = 16X rate 11 = 64X rate	

NOTE:
Baud rate factor applies only if external clock is selected. Factor is 16X if internal clock is selected.

Table 6. MODE REGISTER 2 (MR2)

MR27 – MR24										MR23 – MR20
	TxC	RxC	Pin 8	Pin 20		TxC	RxC	Pin 8	Pin 20	Baud Rate Selection
0000	E	E	TxC	RxC	1000	E	E	NF	RxC/TxC	
0001	E	I	TxC	1X	1001	E	I	TxC	BKDET	
0010	I	E	1X	RxC	1010	I	E	NF	RxC	
0011	I	I	1X	1X	1011	I	I	1X	BKDET	See baud rates in table 1
0100	E	E	TxC	RxC	1100	E	E	NF	RxC/TxC	
0101	E	I	TxC	16X	1101	E	I	TxC	BKDET	
0110	I	E	16X	RxC	1110	I	E	NF	RxC	
0111	I	I	16X	16X	1111	I	I	16X	BKDET	

NOTES:
E = External clock
I = Internal clock (BRG)
NF = No function; output not valid
1X and 16X are clock outputs

Product Specification

Table 7. COMMAND REGISTER (CR)

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode		**Request To Send**	**Reset Error**	**Force Break**	**Receive Control (RxEN)**		**Transmit Control (TxEN)**
00 = Normal operation 01 = Automatic echo mode 10 = Local loopback 11 =Remote loopback		0 = Force $\overline{RTS}$ output high after TxSR serialization 1 = Force $\overline{RTS}$ output low	0 = Normal 1 = Reset error flags in status register (FE, OE, PE)	0 = Normal 1 = Force break	0 = Disable 1 = Enable	Not used. Must be programmed to '1'	0 = Disable 1 = Enable

Table 8. STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
	Data Carrier Detect	**Framing Error**	**Overrun Error**	**Parity Error**	**TxEMT/DSCHG**	**RxRDY**	**TxRDY**
Not used	0 = $\overline{DCD}$ input is high 1 = $\overline{DCD}$ input is low	0 = Normal 1 = Framing Error	0 = Normal 1 = Overrun Error	0 = Normal 1 = Parity error	0 =Normal 1 =Change in $\overline{DCD}$ or transmit shift register is empty	0 = Receive holding register empty 1 = Receive holding register has data	0 = Transmit holding register busy 1 = Transmit holding register busy

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{DCD}$ input (when CR2 or CR0 = 1) or that the transmit shift register has completed transmission of a character and no new character has been loaded into the transmit data holding register. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the transmit data holding register. The DSCHG condition is enabled when the TxEN = 1 or RxEN = 1. It is cleared when the status register is read by the CPU. If the status register is read twice and SR2 = 1 while SR6 remains unchanged, then a TxEMT condition exists. When SR2 is set, the $\overline{INTR}$ output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. This bit is cleared when the receiver is disabled and by a reset error command, CR4.

The overrun error status bit, SR4, indicates that the previous character loaded into the receive holding register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled and by the reset error command, CR4.

Bit SR5 signifies that the received character was not framed by a stop bit; i.e., only the first stop bit is checked. If RHR = 0 when SR5 = 1, a break condition is present. The bit is reset when the receiver is disabled and when the reset error command is given.

SR6 reflects the condition of the DCD input. A low input sets the status bit and a high input clears it.

Product Specification

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Input voltage						V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
Output voltage						V
V_{OL}	Low	I_{OL} = 2.2mA			0.4	
V_{OH}[7]	High	I_{OH} = −400μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to V_{CC}			10	μA
3-state output leakage current						μA
I_{LH}	Data bus high	V_O = 4.0V			10	
I_{LL}	Data bus low	V_O = 0.45V			10	
I_{CC}	Power supply current				150	mA

CAPACITANCE T_A = 25°C, V_{CC} = 0V

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Capacitance						pF
C_{IN}	Input	fc = 1MHz			20	
C_{OUT}	Output	Unmeasured pins tied to ground			20	
$C_{I/O}$	Input/Output[7]				20	

Product Specification

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Pulse width						ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		250			
t_{CED}	CE to CE delay		600			
Set-up and hold time						ns
t_{AS}	Address set-up		10			
t_{AH}	Address hold		10			
t_{CS}	$\overline{R}$/W control set-up		10			
t_{CH}	$\overline{R}$/W control hold		10			
t_{DS}	Data set-up for write		150			
t_{DH}	Data hold for write		10			
t_{RXS}	Rx data set-up		300			
t_{RXH}	Rx data hold		350			
t_{DD}	Data delay time for read	$C_L = 150pF$			200	ns
t_{DF}	Data bus floating time for read	$C_L = 150pF$			100	ns
Input clock frequency						MHz
f_{BRG}[10]	Baud rate generator		1.0	3.6864	4.0	
$f_{R/T}$[10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
Clock state						ns
t_{BRH}[9]	Baud rate high		90			
t_{BRL}[9]	Baud rate low		90			
$t_{R/TH}$	$\overline{TxC}$ or $\overline{RxC}$ high		480			
$t_{R/TL}$[10]	$\overline{TxC}$ or $\overline{RxC}$ low		480			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	$C_L = 150pF$			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output[8]	$C_L = 150pF$		0		ns

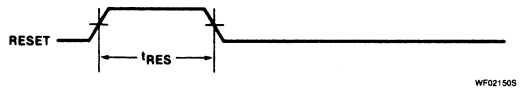
NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and t_{BRL}) and at 0.8V and 2.0V for outputs. Input levels swing between 0.4V and 2.4V, with a transition time of 20ns maximum.
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. $\overline{INTR}$ output is open drain.
8. Parameter applies when internal transmitter clock is used.
9. t_{BRH} and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. In asynchronous local loopback mode, using 1X clock, the following parameters apply:
    $f_{R/T} = 0.83MHz$ max
    $t_{R/TL} = 700ns$ min

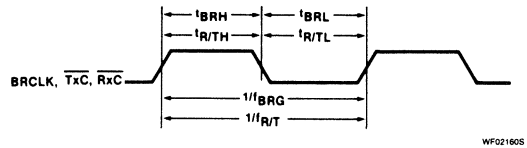
Product Specification

TIMING DIAGRAMS

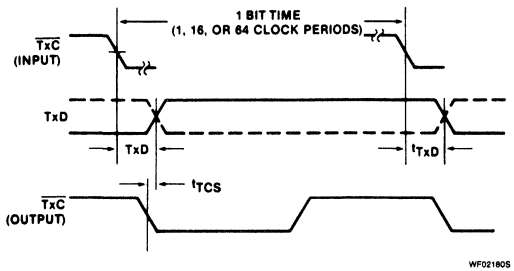
RESET



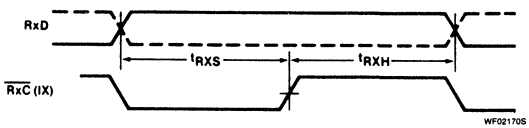
CLOCK



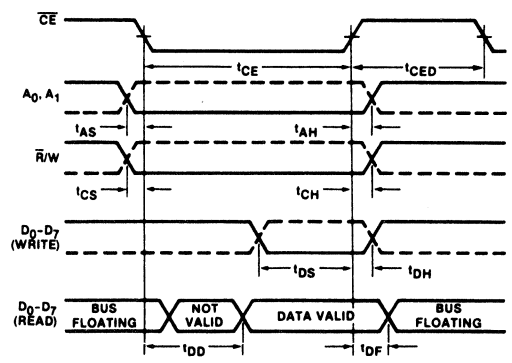
TRANSMIT



RECEIVE



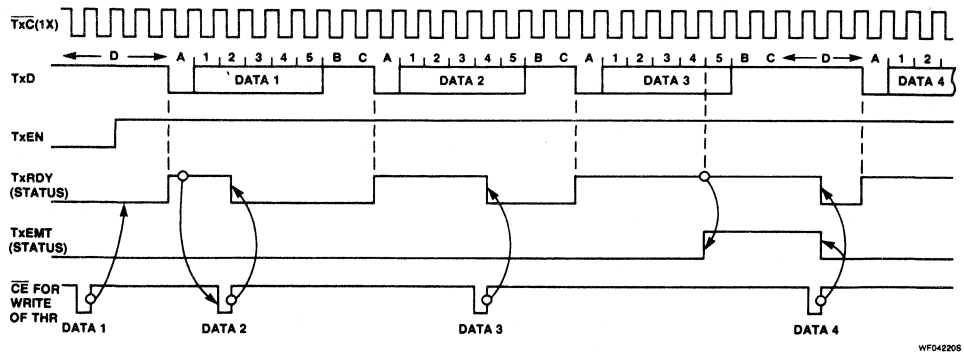
READ AND WRITE



Product Specification

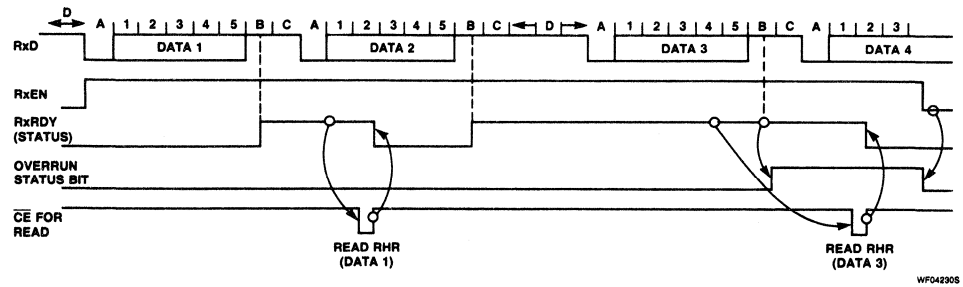
TIMING DIAGRAMS (Continued)

TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits)



NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

RxRDY (Shown for 5-bit characters, no parity, 2 stop bits)

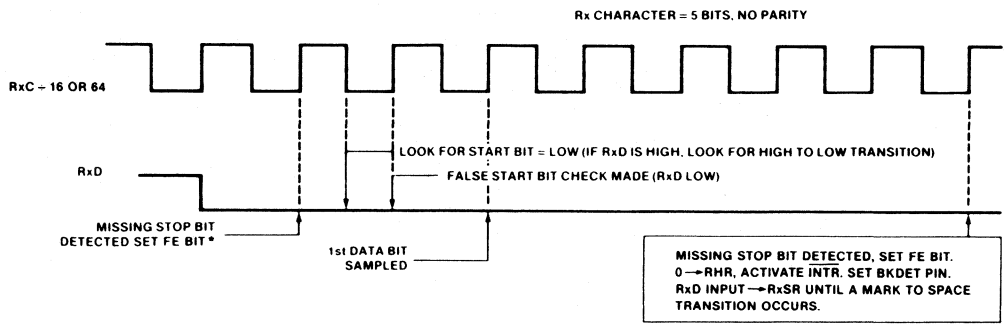


NOTES:
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition
Only one stop bit is detected.

Product Specification

TIMING DIAGRAMS (Continued)

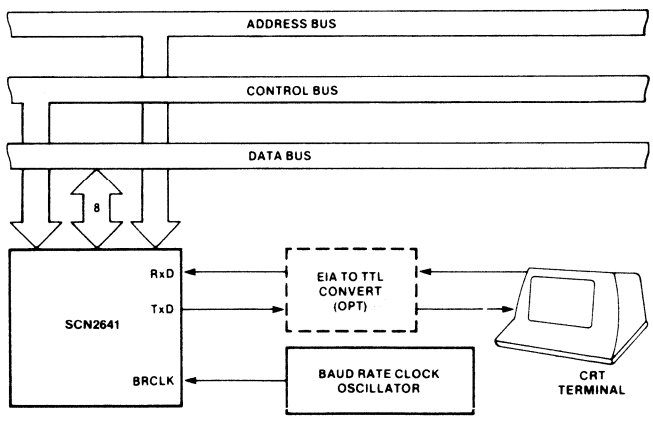
BREAK DETECTION TIMING



*If the stop bit is present, the start bit search will commence immediately.

TYPICAL APPLICATIONS

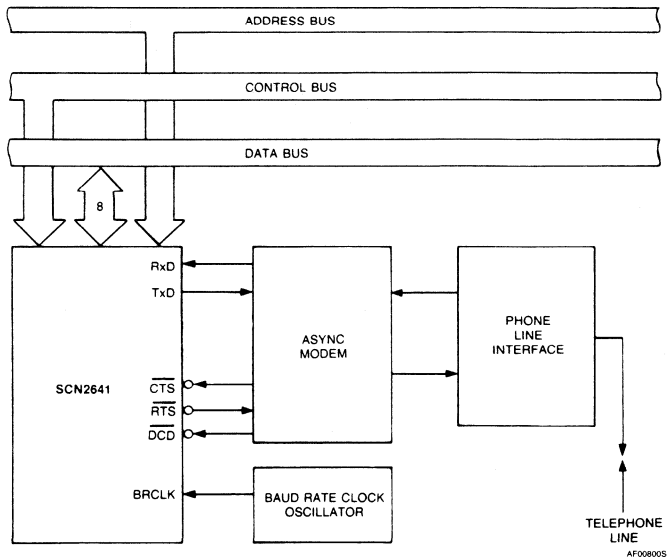
ASYNCHRONOUS INTERFACE TO CRT TERMINAL



Product Specification

TYPICAL APPLICATIONS (Continued)

ASYNCHRONOUS INTERFACE TO TELEPHONE LINES



PROGRAMMABLE COMMUNICATIONS INTERFACE (PCI)

Originally published by Signetics May 1983

DESCRIPTION

The Signetics SCN2651 PCI is a universal synchronous/asynchronous data communications controller chip designed for microcomputer systems. It interfaces directly to the Signetics SCN2650 microprocessor and may be used in a polled or interrupt driven system environment. The SCN2651 accepts programmed instructions from the microprocessor and supports many serial data communication disciplines, synchronous and asynchronous, in the full or half-duplex mode.

The PCI serializes parallel data characters received from the microprocessor for transmission. Simultaneously, it can receive serial data and convert it into parallel data characters for input to the microcomputer.

The SCN2651 contains a baud rate generator which can be programmed to either accept an external clock or to generate internal transmit or receive clocks. Sixteen different baud rates can be selected under program control when operating in the internal clock mode.

The PCI is constructed using Signetics n-channel silicon gate depletion load technology and is packaged in a 28-pin DIP.

FEATURES

- **Synchronous operation**
  - **5 to 8-bit characters**
  - **Single or double SYN operation**
  - **Internal character synchronization**
  - **Transparent or non-transparent mode**
  - **Automatic SYN or DLE-SYN insertion**
  - **SYN or DLE stripping**
  - **Odd, even, or no parity**
  - **Local or remote maintenance loop back mode**
  - **Baud rate: dc to 1M bps (1X clock)**
- **Asynchronous operation**
  - **5 to 8-bit characters**
  - **1, 1 1/2 or 2 stop bits**
  - **Odd, even, or no parity**
  - **Parity, overrun and framing error detection**
  - **Line break detection and generation**
  - **False start bit detection**
  - **Automatic serial echo mode**
  - **Local or remote maintenance loop back mode**
  - **Baud rate: dc to 1M bps (1X clock)**
    - **dc to 62.5K bps (16X clock)**
    - **dc to 15.625K bps (64X clock)**

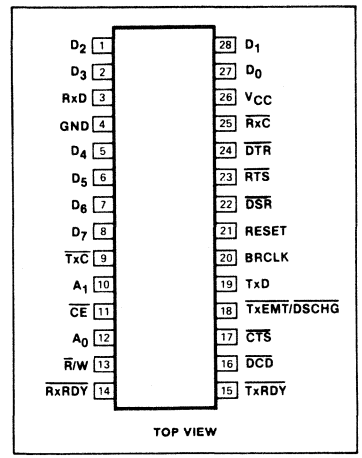
OTHER FEATURES

- **Internal or external baud rate clock**
- **16 internal rates-50 to 19,200 baud**
- **Double buffered transmitter and receiver**
- **Full or half duplex operation**
- **Fully compatible with 2650 CPU**
- **TTL compatible inputs and outputs**
- **Single 5V power supply**
- **No system clock required**
- **28-pin dual in-line package**

APPLICATIONS

- **Intelligent terminals**
- **Network processors**
- **Front end processors**
- **Remote data concentrators**
- **Computer to computer links**
- **Serial peripherals**

PIN CONFIGURATION



TOP VIEW

PIN DESIGNATION

PIN NO.	SYMBOL	NAME AND FUNCTION	TYPE
27,28,1,2, 5-8	D_0-D_7	8-bit data bus	I/O
21	RESET	Reset	I
12,10	A_0-A_1	Internal register select lines	I
13	$\overline{R}/W$	Read or write command	I
11	$\overline{CE}$	Chip enable input	I
22	$\overline{DSR}$	Data set ready	I
24	$\overline{DTR}$	Data terminal ready	O
23	$\overline{RTS}$	Request to send	O
17	$\overline{CTS}$	Clear to send	I
16	$\overline{DCD}$	Data carrier detected	I
18	$\overline{TxEMT/DSCHG}$	Transmitter empty or data set change	O
9	$\overline{TxC}$	Transmitter clock	I/O
25	$\overline{RxC}$	Receiver clock	I/O
19	TxD	Transmitter data	O
3	RxD	Receiver data	I
15	$\overline{TxRDY}$	Transmitter ready	O
14	$\overline{RxRDY}$	Receiver ready	O
20	BRCLK	Baud rate generator clock	I
26	V_{CC}	+5V supply	I
4	GND	Ground	I

ORDERING CODE

PACKAGES	$V_{CC} = 5V \pm 5\%$		
	COMMERCIAL	AUTOMOTIVE	MILITARY
	0°C to +70°C	−40°C to +85°C	−55°C to +125°C
Ceramic DIP	SCN2651CC1I28	SCN2651CA1I28	SCN2651CM1I28
Plastic DIP	SCN2651CC1N28	Contact Factory	Not Available

Table 1 BAUD RATE GENERATOR CHARACTERISTICS CRYSTAL FREQUENCY = 5.0688MHz

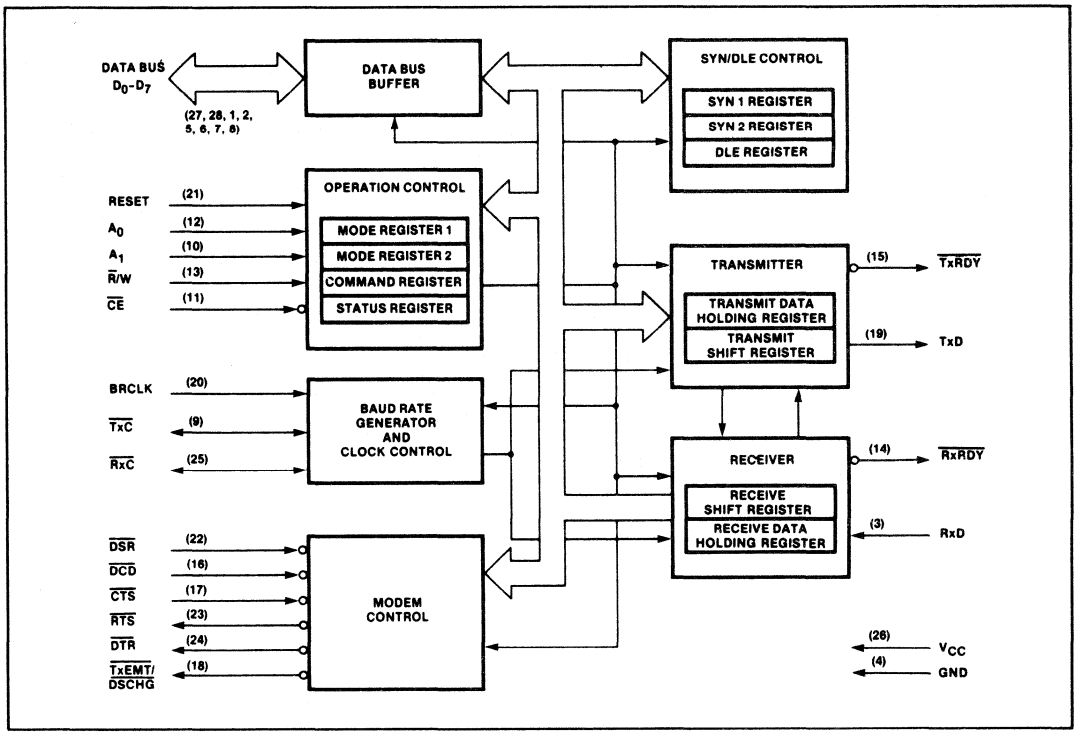
BAUD RATE	THEORETICAL FREQUENCY 16X CLOCK	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
50	0.8 KHz	0.8 KHz	--	6336
75	1.2	1.2	--	4224
110	1.76	1.76	--	2880
134.5	2.152	2.1523	0.016	2355
150	2.4	2.4	--	2112
300	4.8	4.8	--	1056
600	9.6	9.6	--	528
1200	19.2	19.2	--	264
1800	28.8	28.8	--	176
2000	32.0	32.081	0.253	158
2400	38.4	38.4	--	132
3600	57.6	57.6	--	88
4800	76.8	76.8	--	66
7200	115.2	115.2	--	44
9600	153.6	153.6	--	33
19200 *	307.2	316.8	3.125	16

NOTE
*Error at 19200 can be reduced to zero by using crystal frequency 4.9152MHz
16X clock is used in asynchronous mode. In synchronous mode, clock multiplier is 1X.

Table 2 CPU-RELATED SIGNALS

PIN NAME	PIN NO.	INPUT/OUTPUT	FUNCTION
V_{CC}	26	I	+5V supply input
GND	4	I	Ground
RESET	21	I	A high on this input performs a master reset on the SCN2651. This signal asynchronously terminates any device activity and clears the Mode, Command and Status registers. The device assumes the idle state and remains there until initialized with the appropriate control words.
A_1-A_0	10,12	I	Address lines used to select internal PCI registers.
$\overline{R}/W$	13	I	Read command when low, write command when high.
$\overline{CE}$	11	I	Chip enable command. When low, indicates that control and data lines to the PCI are valid and that the operation specified by the $\overline{R}/W$, A_1 and A_0 inputs should be performed. When high, places the D_0-D_7 lines in the tri-state condition.
D_7-D_0	8,7,6,5, 2,1,28,27	I/O	8-bit, three-state data bus used to transfer commands, data and status between PCI and the CPU. D_0 is the least significant bit; D_7 the most significant bit.
$\overline{TxRDY}$	15	O	This output is the complement of Status Register bit SR0. When low, it indicates that the Transmit Data Holding Register (THR) is ready to accept a data character from the CPU. It goes high when the data character is loaded. This output is valid only when the transmitter is enabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{RxRDY}$	14	O	This output is the complement of Status Register bit SR1. When low, it indicates that the Receive Data Holding Register (RHR) has a character ready for input to the CPU. It goes high when the RHR is read by the CPU, and also when the receiver is disabled. It is an open drain output which can be used as an interrupt to the CPU.
$\overline{TxEMT}/\overline{DSCHG}$	18	O	This output is the complement of Status Register bit SR2. When low, it indicates that the transmitter has completed serialization of the last character loaded by the CPU, or that a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs has occurred. This output goes high when the Status Register is read by the CPU, if the TxEMT condition does not exist. Otherwise, the THR must be loaded by the CPU for this line to go high. It is an open drain output which can be used as an interrupt to the CPU.

BLOCK DIAGRAM



BLOCK DIAGRAM

The PCI consists of six major sections. These are the transmitter, receiver, timing, operation control, modem control and SYN/DLE control. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains Mode Registers 1 and 2, the Command Register, and the Status Register. Details of register addressing and protocol are presented in the PCI Programming section of this data sheet.

Timing

The PCI contains a Baud Rate Generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 commonly used baud rates, any one of which can be selected for full duplex operation. See Table 1.

Receiver

The Receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for bits or characters that are unique to the communication technique and sends an "assembled" character to the CPU.

Transmitter

The Transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate characters or bits (based on the communication technique) and outputs a composite serial stream of data on the TxD output pin.

Modem Control

The modem control section provides interfacing for three input signals and three output signals used for "handshaking" and status indication between the CPU and a modem.

SYN/DLE Control

This section contains control circuitry and three 8-bit registers storing the SYN1, SYN2, and DLE characters provided by the CPU. These registers are used in the synchronous mode of operation to provide the characters required for synchronization, idle fill and data transparency.

INTERFACE SIGNALS

The PCI interface signals can be grouped into two types: the CPU-related signals (shown in Table 2), which interface the 2651 to the microprocessor system, and the device-related signals (shown in Table 3), which are used to interface to the communications device or system.

Table 3 DEVICE-RELATED SIGNALS

PIN NAME	PIN NO.	INPUT/OUTPUT	FUNCTION
BRCLK	20	I	5.0688MHz clock input to the internal baud rate generator. Not required if external receiver and transmitter clocks are used.
$\overline{RxC}$	25	I/O	Receiver clock. If external receiver clock is programmed, this input controls the rate at which the character is to be received. Its frequency is 1X, 16X or 64X the baud rate, as programmed by Mode Register 1. Data is sampled on the rising edge of the clock. If internal receiver clock is programmed, this pin becomes an output at 1X the programmed baud rate.*
$\overline{TxC}$	9	I/O	Transmitter clock. If external transmitter clock is programmed, this input controls the rate at which the character is transmitted. Its frequency is 1X, 16X or 64X the baud rate, as programmed by Mode Register 1. The transmitted data changes on the falling edge of the clock. If internal transmitter clock is programmed, this pin becomes an output at 1X the programmed baud rate.*
RxD	3	I	Serial data input to the receiver. "Mark" is high, "Space" is low.
TxD	19	O	Serial data output from the transmitter. "Mark" is high, "Space" is low. Held in Mark condition when the transmitter is disabled.
$\overline{DSR}$	22	I	General purpose input which can be used for Data Set Ready or Ring Indicator condition. Its complement appears as Status Register bit SR7. Causes a low output on $\overline{TxEMT/DSCHG}$ when its state changes.
$\overline{DCD}$	16	I	Data Carrier Detect input. Must be low in order for the receiver to operate. Its complement appears as Status Register bit SR6. Causes a low output on $\overline{TxEMT/DSCHG}$ when its state changes.
$\overline{CTS}$	17	I	Clear to Send input. Must be low in order for the transmitter to operate. If it goes high during transmission, the character in the Transmit Shift Register will be transmitted before termination.
$\overline{DTR}$	24	O	General purpose output which is the complement of Command Register bit CR1. Normally used to indicate Data Terminal Ready.
$\overline{RTS}$	23	O	General purpose output which is the complement of Command Register bit CR5. Normally used to indicate Request to Send.

NOTE
*$\overline{RxC}$ and $\overline{TxC}$ outputs have short circuit protection max. C_L 100pf

OPERATION

The functional operation of the SCN2651 is programmed by a set of control words supplied by the CPU. These control words specify items such as synchronous or asynchronous mode, baud rate, number of bits per character, etc. The programming procedure is described in the PCI Programming section of this data sheet.

After programming, the PCI is ready to perform the desired communications functions. The receiver performs serial to parallel conversion of data received from a modem or equivalent device. The transmitter converts parallel data received from the CPU to a serial bit stream. These actions are accomplished within the framework specified by the control words.

Receiver

The SCN2651 is conditioned to receive data when the $\overline{DCD}$ input is low and the RxEN bit in the command register is true. In the asynchronous mode, the receiver looks for a high to low transition of the start bit on the RxD input line. If a transition is detected, the state of the RxD line is sampled again after a delay of one-half of a bit time. If RxD is now high, the search for a valid start bit is begun again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input line at one bit time intervals until the proper number of data bits, the parity bit, and the stop bit(s) have been assembled. The data is then transferred to the Receive Data Holding Register, the RxRDY bit in the status register is set, and the $\overline{RxRDY}$ output is asserted. If the character length is less than 8 bits, the high order unused bits in the Holding Register are set to zero. The Parity Error, Framing Error, and Overrun Error status bits are strobed into the status register on the positive going edge of $\overline{RxC}$ corresponding to the received character boundary. If a break condition is detected (RxD is low for the entire character as well as the stop bit(s)), only one character consisting of all zeros (with the FE status bit set) will be transferred to the Holding Register. The RxD input must return to a high condition before a search for the next start bit begins.

When the PCI is initialized into the synchronous mode, the receiver first enters the hunt mode on a 0 to 1 transition of RxEN (CR2). In this mode, as data is shifted into the Receiver Shift Register a bit at a time, the contents of the register are compared to the contents of the SYN1 register. If the two are not equal, the next bit is shifted in and the comparison is repeated. When the two registers match, the hunt mode is terminated and character assembly mode begins. If single SYN operation is programmed, the SYN DETECT status bit is set. If double SYN operation is programmed, the first character assembled after SYN1 must be SYN2 in order for the SYN DETECT bit to be set. Otherwise, the PCI returns to the hunt mode. (Note that the sequence SYN1-SYN1-SYN2 will not achieve synchronization). When synchronization has been achieved, the PCI continues to assemble characters and transfer them to the Holding Register, setting the RxRDY status bit and asserting the $\overline{RxRDY}$ output each time a character is transferred. The PE and OE status bits are set as appropriate. Further receipt of the appropriate SYN sequence sets the SYN DETECT status bit. If the SYN stripping mode is commanded, SYN characters are not transferred to the Holding Register. Note that the SYN characters used to establish initial synchronization are not transferred to the Holding Register in any case.

Transmitter

The PCI is conditioned to transmit data when the $\overline{CTS}$ input is low and the TxEN command register bit is set. The SCN2651 indicates to the CPU that it can accept a character for transmission by setting the

TxRDY status bit and asserting the $\overline{\text{TxRDY}}$ output. When the CPU writes a character into the Transmit Data Holding Register, these conditions are negated. Data is transferred from the Holding Register to the Transmit Shift Register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again. Thus, one full character time of buffering is provided.

In the asynchronous mode, the transmitter automatically sends a start bit followed by the programmed number of data bits, the least significant bit being sent first. It then appends an optional odd or even parity bit and the programmed number of stop bits. If, following transmission of the data bits, a new character is not available in the Transmit Holding Register, the TxD output remains in the marking (high) condition and the $\overline{\text{TxEMT}}/\overline{\text{DSCHG}}$ output and its corresponding status bit are asserted. Transmission resumes when the CPU loads a new character into the Holding Register. The transmitter can be forced to output a continuous low (BREAK) condition by setting the Send Break command bit high.

In the synchronous mode, when the SCN2651 is initially conditioned to transmit, the TxD output remains high and the TxRDY condition is asserted until the first character to be transmitted (usually a SYN character) is loaded by the CPU. Subsequent to this, a continuous stream of characters is transmitted. No extra bits (other than parity, if commanded) are generated by the PCI unless the CPU fails to send a new character to the PCI by the time the transmitter has completed sending the previous character.

Since synchronous communication does not allow gaps between characters, the PCI asserts TxEMT and automatically "fills" the gap by transmitting SYN1s, SYN1-SYN2 doublets, or DLE-SYN1 doublets, depending on the state of MR16 and MR17. Normal transmission of the message resumes when a new character is available in the Transmit Data Holding Register. If the SEND DLE bit in the command register is true, the DLE character is automatically transmitted prior to transmission of the message character in THR.

PCI PROGRAMMING

Prior to initiating data communications, the SCN2651 operational mode must be programmed by performing write operations to the mode and command registers. In addition, if synchronous operation is programmed, the appropriate SYN/DLE registers must be loaded. The PCI can be reconfigured at any time during program execution. However, if the change has an effect on the reception of a character the receiver should be disabled. Alternatively if

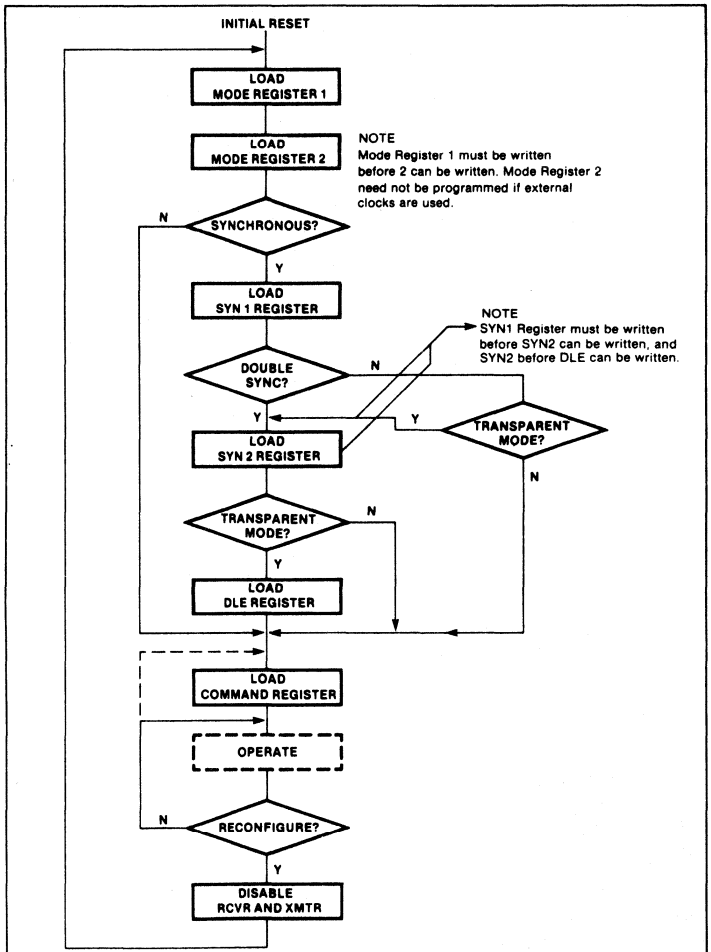


Figure 1. 2651 Initialization Flow Chart

Table 4 2651 REGISTER ADDRESSING

$\overline{\text{CE}}$	A_1	A_0	$\overline{\text{R}}/\text{W}$	FUNCTION
1	X	X	X	Tri-state data bus
0	0	0	0	Read receive holding register
0	0	0	1	Write transmit holding register
0	0	1	0	Read status register
0	0	1	1	Write SYN1/SYN2/DLE registers
0	1	0	0	Read mode registers 1/2
0	1	0	1	Write mode registers 1/2
0	1	1	0	Read command register
0	1	1	1	Write command register

NOTE
See AC Characteristics section for timing requirements.

the change is made 1½ RxC periods after RxRDY goes active it will affect the next character assembly. A flowchart of the initialization process appears in Figure 1.

The internal registers of the PCI are accessed by applying specific signals to the $\overline{CE}$, $\overline{R}/W$, A_1 and A_0 inputs. The conditions necessary to address each register are shown in Table 4.

The SYN1, SYN2, and DLE registers are accessed by performing write operations with the conditions $A_1 = 0$, $A_0 = 1$, and $\overline{R}/W = 1$. The first operation loads the SYN1 register. The next loads the SYN2 register, and the third loads the DLE register. Reading or loading the mode registers is done in a similar manner. The first write (or read) operation addresses Mode Register 1, and a subsequent operation addresses Mode Register 2. If more than the required number of accesses are made, the internal sequencer recycles to point at the first register. The pointers are reset to SYN1 Register and Mode Register 1 by a RESET input or by performing a "Read Command Register" operation, but are unaffected by any other read or write operation.

The SCN2651 register formats are summarized in Tables 5, 6, 7 and 8. Mode Registers 1 and 2 define the general operational characteristics of the PCI, while the Command Register controls the operation within this basic frame-work. The PCI indicates its status in the Status Register. these registers are cleared when a RESET input is applied.

Mode Register 1 (MR1)

Table 5 illustrates Mode Register 1. Bits MR11 and MR10 select the communication format and baud rate multiplier. 00 specifies synchronous mode and 1X multiplier. 1X, 16X, and 64X multipliers are programmable for asynchronous format. However, the multiplier in asynchronous format applies only if the external clock input option is selected by MR24 or MR25.

MR13 and MR12 select a character length of 5, 6, 7, or 8 bits. The character length does not include the parity bit, if programmed, and does not include the start and stop bits in asynchronous mode.

MR14 controls parity generation. If enabled, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR15 selects odd or even parity when parity is enabled by MR14.

In asychronous mode, MR17 and MR16 select character framing of 1, 1.5, or 2 stop bits. (If 1X baud rate is programmed, 1.5 stop bits defaults to 1 stop bits on transmit). In synchronous mode, MR17 controls the number of SYN characters used to establish synchronization and for character fill when the transmitter is idle. SYN1 alone is used if MR17 = 1, and SYN1-SYN2 is used when MR17 = 0. If the transparent mode is specified by MR16, DLE-SYN1 is used for character fill and SYN Detect, but the normal synchronization sequence is used. Also DLE stripping and DLE Detect (with MR14 = 0) are enabled.

Mode Register 2 (MR2)

Table 6 illustrates Mode Register 2. MR23, MR22, MR21, and MR20 control the frequency of the internal baud rate generator (BRG). Sixteen rates are selectable. When driven by a 5.0688 MHz input at the BRCLK input (pin 20), the BRG output has zero error except at 134.5, 2000, and 19,200 baud, which have errors of +0.016%, +0.235%, and +3.125% respectively.

MR25 and MR24 select either the BRG or the external inputs $\overline{TxC}$ and $\overline{RxC}$ as the clock source for the transmitter and receiver, respectively. If the BRG clock is selected,

Table 5 MODE REGISTER 1 (MR1)

MR17	MR16	MR15	MR14	MR13	MR12	MR11	MR10
		Parity Type	Parity Control	Character Length		Mode and Baud Rate Factor	
ASYNCH: STOP BIT LENGTH 00 = INVALID 01 = 1 STOP BIT 10 = 1½ STOP BITS 11 = 2 STOP BITS		0 = ODD 1 = EVEN	0 = DISABLED 1 = ENABLED	00 = 5 BITS 01 = 6 BITS 10 = 7 BITS 11 = 8 BITS		00 = SYNCHRONOUS 1X RATE 01 = ASYNCHRONOUS 1X RATE 10 = ASYNCHRONOUS 16X RATE 11 = ASYNCHRONOUS 64X RATE	
SYNCH: NUMBER OF SYN CHAR 0 = DOUBLE SYN 1 = SINGLE SYN	SYNCH: TRANSPARENCY CONTROL 0 = NORMAL 1 = TRANSPARENT						

NOTE
Baud rate factor in asynchronous applies only if external clock is selected. Factor is 16X if internal clock is selected. Mode must be selected (MR11, MR10) in any case.

Table 6 MODE REGISTER 2 (MR2)

MR27	MR26	MR25	MR24	MR23	MR22	MR21	MR20
		Transmitter Clock	Receiver Clock	Baud Rate Selection			
NOT USED		0 = EXTERNAL 1 = INTERNAL	0 = EXTERNAL 1 = INTERNAL	0000 = 50 BAUD 0001 = 75 0010 = 110 0011 = 134.5 0100 = 150 0101 = 300 0110 = 600 0111 = 1200		1000 = 1800 BAUD 1001 = 2000 1010 = 2400 1011 = 3600 1100 = 4800 1101 = 7200 1110 = 9600 1111 = 19,200	

the baud rate factor in asynchronous mode is 16X regardless of the factor selected by MR11 and MR10. In addition, the corresponding clock pin provides an output at 1X the baud rate.

Command Register (CR)

Table 7 illustrates Command Register. Bits CR0 (TxEN) and CR2 (RxEN) enable or disable the transmitter and receiver respectively. A 0 to 1 transition of CR2 forces start bit search (async mode) or hunt mode (sync mode) on the second $\overline{RxC}$ rising edge. Disabling the receiver causes $\overline{RxRDY}$ to go high (inactive). If the transmitter is disabled, it will complete the transmission of the character in the Transmit Shift Register (if any) prior to terminating operation. The TxD output will then remain in the marking state (high) while the $\overline{TxRDY}$ and $\overline{TxEMT}$ will go high (inactive). If the receiver is disabled, it will terminate operation immediately. Any character being assembled will be neglected.

Bits CR1 (DTR) and CR5 (RTS) control the $\overline{DTR}$ and $\overline{RTS}$ outputs. Data at the outputs is the logical complement of the register data.

In asynchronous mode, setting CR3 will force and hold the TxD output low (spacing condition) at the end of the current transmitted character. Normal operation resumes when CR3 is cleared. The TxD line will go high for a least one bit time before beginning transmission of the next character in the Transmit Data Holding Register. In synchronous mode, setting CR3 causes the transmission of the DLE register contents prior to sending the character in the Transmit Data Holding Register. CR3 should be reset in response to the next TxRDY.

Setting CR4 causes the error flags in the Status Register (SR3, SR4, and SR5) to be cleared. This is a one time command. There is no internal latch for this bit.

The PCI can operate in one of four submodes within each major mode (synchronous or asynchronous). The operational sub-mode is determined by CR7 and CR6. CR7-CR6 = 00 is the normal mode, with the transmitter and receiver operating independently in accordance with the Mode and Status Register instructions.

In asynchronous mode, CR7-CR6 = 01 places the PCI in the Automatic Echo mode. Clocked, regenerated received data is automatically directed to the TxD line while normal receiver operation continues. The receiver must be enabled (CR2 = 1), but the transmitter need not be enabled. CPU to receiver communications continues normally, but the CPU to transmitter link is disabled. Only the first character of a break condition is echoed. The TxD output will go high until the next valid start is detected. The following conditions are true while in Automatic Echo mode:

1. Data assembled by the receiver is automatically placed in the Transmit Holding Register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. $\overline{TxRDY}$ output = 1.
4. The $\overline{TxEMT/DSCHG}$ pin will reflect only the data set change condition.
5. The TxEN command (CR0) is ignored.

In synchronous mode, CR7-CR6 = 01 places the PCI in the Automatic SYN/DLE Stripping mode. The exact action taken depends on the setting of bits MR17 and MR16:

1. In the non-transparent, single SYN mode (MR17-MR16 = 10), characters in the data stream matching SYN1 are not transferred to the Receive Data Holding Register (RHR).
2. In the non-transparent, double SYN mode (MR17-MR16 = 00), characters in the data stream matching SYN1, or SYN2 if immediately preceded by SYN1, are not transferred to the RHR. However, only the first SYN1 of an SYN1-SYN1 pair is stripped.
3. In transparent mode (MR16 =1), characters in the data stream matching DLE, or SYN1 if immediately preceded by DLE, are not transferred to the RHR. However, only the first DLE of a DLE-DLE pair is stripped.

Note that Automatic Stripping mode does not affect the setting of the DLE Detect and SYN Detect status bits (SR3 and SR5).

Two diagnostic sub-modes can also be configured. In Local Loop Back mode (CR7-CR6 = 10), the following loops are connected internally:

1. The transmitter output is connected to the receiver input.
2. $\overline{DTR}$ is connected to $\overline{DCD}$ and $\overline{RTS}$ is connected to $\overline{CTS}$.
3. The receiver is clocked by the transmit clock.
4. The $\overline{DTR}$, $\overline{RTS}$ and $\overline{TxD}$ outputs are held high.
5. The $\overline{CTS}$, $\overline{DCD}$, $\overline{DSR}$ and RxD inputs are ignored.

Additional requirements to operate in the Local Loop Back mode are that CR0 (TxEN), CR1 (DTR), and CR5 (RTS) must be set to 1. CR2 (RxEN) is ignored by the PCI.

The second diagnostic mode is the Remote Loop Back mode (CR7-CR6 = 11). In this mode:

1. Data assembled by the receiver is automatically placed in the Transmit Holding Register and retransmitted by the transmitter on the TxD output.
2. The transmitter is clocked by the receive clock.
3. No data is sent to the local CPU, but the error status conditions (PE, OE, FE) are set.
4. The $\overline{RxRDY}$, $\overline{TxRDY}$, and $\overline{TxEMT/DSCHG}$ outputs are held high.
5. CR1 (TxEN) is ignored.
6. All other signals operate normally.

Status Register

The data contained in the Status Register (as shown in Table 8) indicate receiver and transmitter conditions and modem/data set status.

SR0 is the Transmitter Ready (TxRDY) status bit. It, and its corresponding output, are valid only when the transmitter is enabled. If equal to 0, it indicates that the Transmit Data Holding Register has been loaded by the CPU and the data has not been transferred to the Transmit Shift Register. If set equal to 1, it indicates that the Holding Register is ready to accept data from the CPU. This bit is initially set when the Transmitter is enabled by CR0, unless a character

Table 7 COMMAND REGISTER (CR)

CR7	CR6	CR5	CR4	CR3	CR2	CR1	CR0
Operating Mode		Request to Send	Reset Error		Receive Control (RxEN)	Data Terminal Ready	Transmit Control (TxEN)
00 = NORMAL OPERATION 01 = ASYNCH: AUTOMATIC ECHO MODE SYNCH: SYN AND/OR DLE STRIPPING MODE 10 = LOCAL LOOP BACK 11 = REMOTE LOOP BACK		0 = FORCE $\overline{RTS}$ OUTPUT HIGH 1 = FORCE $\overline{RTS}$ OUTPUT LOW	0 = NORMAL 1 = RESET ERROR FLAG IN STATUS REG (FE, OE, PE/DLE DETECT)	**ASYNCH: FORCE BREAK** 0 = NORMAL 1 = FORCE BREAK **SYNCH: SEND DLE** 0 = NORMAL 1 = SEND DLE	0 = DISABLE 1 = ENABLE	0 = FORCE $\overline{DTR}$ OUTPUT HIGH 1 = FORCE $\overline{DTR}$ OUTPUT LOW	0 = DISABLE 1 = ENABLE

Table 8 STATUS REGISTER (SR)

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
Data Set Ready	**Data Carrier Detect**	**FE/SYN Detect**	**Overrrun**	**PE/DLE Detect**	**TxEMT/DSCHG**	**RxRDY**	**TxRDY**
0 = $\overline{DSR}$ INPUT IS HIGH 1 = $\overline{DSR}$ INPUT IS LOW	0 = $\overline{DCD}$ INPUT IS HIGH 1 = $\overline{DCD}$ INPUT IS LOW	**ASYNCH:** 0 = NORMAL 1 = FRAMING ERROR **SYNCH:** 0 = NORMAL 1 = SYN CHAR DETECTED	0 = NORMAL 1 = OVERRUN ERROR	**ASYNCH:** 0 = NORMAL 1 = PARITY ERROR **SYNCH:** 0 = NORMAL 1 = PARITY ERROR OR DLE CHAR RECEIVED	0 = NORMAL 1 = CHANGE IN $\overline{DSR}$ OR $\overline{DCD}$, OR TRANSMIT SHIFT REGISTER IS EMPTY	0 = RECEIVE HOLDING REG EMPTY 1 = RECEIVE HOLDING REG HAS DATA	0 = TRANSMIT HOLDING REG BUSY 1 = TRANSMIT HOLDING REG EMPTY

has previously been loaded into the Holding Register. It is not set when the Automatic Echo or Remote Loop Back modes are programmed. When this bit is set, the $\overline{TxRDY}$ output pin is low. In the Automatic Echo and Remote Loop Back modes, the output is held high.

SR1, the Receiver Ready (RxRDY) status bit, indicates the condition of the Receive Data Holding Register. If set, it indicates that a character has been loaded into the Holding Register from the Receive Shift Register and is ready to be read by the CPU. If equal to zero, there is no new character in the Holding Register. This bit is cleared when the CPU reads the Receive Data Holding Register or when the receiver is disabled by CR2. When set, the $\overline{RxRDY}$ output is low.

The TxEMT/DSCHG bit, SR2, when set, indicates either a change of state of the $\overline{DSR}$ or $\overline{DCD}$ inputs or that the Transmit Shift Register has completed transmission of a character and no new character has been loaded into the Transmit Data Holding Register. Note that in synchronous mode this bit will be set even though the appropriate "fill" character is transmitted. TxEMT will not go active until at least one character has been transmitted. It is cleared by loading the Transmit Data Holding Register. The DSCHG condition is enabled when TxEN=1 or RxEN=1. It is cleared when the Status Register is read by the CPU. When SR2 is set, the TxEMT/DSCHG output is low.

SR3, when set, indicates a received parity error when parity is enabled by MR14. In synchronous transparent mode (MR16 = 1), with parity disabled, it indicates that a character matching the DLE Register has been received. However, only the first DLE of two successive DLEs will set SR3. This bit is cleared when the receiver is disabled and by the Reset Error command, CR4.

The Overrun Error status bit, SR4, indicates that the previous character loaded into the Receive Holding Register was not read by the CPU at the time a new received character was transferred into it. This bit is cleared when the receiver is disabled and by the Reset Error command, CR4.

In asynchronous mode, bit SR5 signifies that the received character was not framed by the programmed number of stop bits. (If 1.5 stop bits are programmed, only the first stop bit is checked.) If RHR = 0 when SR5 = 1 a break condition is present. In synchronous non-transparent mode (MR16 = 0), it indicates receipt of the SYN1 character is single SYN mode or the SYN1-SYN2 pair in double SYN mode. In synchronous transparent mode (MR16 = 1), this bit is set upon detection of the initial synchronizing characters (SYN1 or SYN1-SYN2) and, after synchronization has been achieved, when a DLE-SYN1 pair is received. The bit is reset when the receiver is disabled, when the Reset Error command is given in asynchronous mode, and when the Status Register is read by the CPU in the synchronous mode.

SR6 and SR7 reflect the conditions of the $\overline{DCD}$ and $\overline{DSR}$ inputs respectively. A low input sets its corresponding status bit and a high input clears it.

DC ELECTRICAL CHARACTERISTICS[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Input voltage					V
V_{IL}	Low				0.8	
V_{IH}	High		2.0			
	Output voltage					V
V_{OL}	Low	I_{OL} = 1.6mA			0.4	
V_{OH}	High	I_{OH} = -100μA	2.4			
I_{IL}	Input leakage current	V_{IN} = 0 to 5.25V	-10		10	μA
Tristate	Output leakage current					μA
I_{LH}	Data bus high	V_O = 4.0V	-10		10	
I_{LL}	Data bus low	V_O = 0.45V	-10		10	
I_{CC}	Power supply current				150	mA

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	Note 4	°C
Storage temperature	-65 to +150	°C
All voltages with respect to ground[3]	-0.5 to +6.0	V

AC ELECTRICAL CHARACTERISTICS[4,5,6]

	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Pulse width					ns
t_{RES}	Reset		1000			
t_{CE}	Chip enable		300			
	Setup and hold time					ns
t_{AS}	Address setup		20			
t_{AH}	Address hold		20			
t_{CS}	$\overline{R}$/W control setup		20			
t_{CH}	$\overline{R}$/W control hold		20			
t_{DS}	Data setup for write		225			
t_{DH}	Data hold for write		0			
t_{RXS}	Rx data setup		300			
t_{RXH}	Rx data hold		350			
t_{DD}	Data delay time for read	C_L = 100pF			250	ns
t_{DF}	Data bus floating time for read	C_L = 100pF			150	ns
t_{CED}	$\overline{CE}$ to $\overline{CE}$ delay		700			ns
	Input clock frequency					MHz
f_{BRG}	Baud rate generator		1.0	5.0688	5.0738	
$f_{R/T}$[10]	$\overline{TxC}$ or $\overline{RxC}$		dc		1.0	
	Clock width					ns
t_{BRH}[9]	Baud rate high		70			
t_{BRL}[9]	Baud rate low		70			
$t_{R/TH}$	$\overline{TxC}$ or $\overline{RxC}$ high		500			
$t_{R/TL}$[10]	$\overline{TxC}$ or $\overline{RxC}$ low		500			
t_{TXD}	TxD delay from falling edge of $\overline{TxC}$	C_L = 100pF			650	ns
t_{TCS}	Skew between TxD changing and falling edge of $\overline{TxC}$ output[8]	C_L = 100pF		0		ns

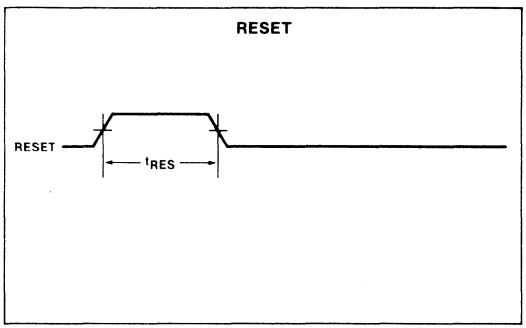
NOTES

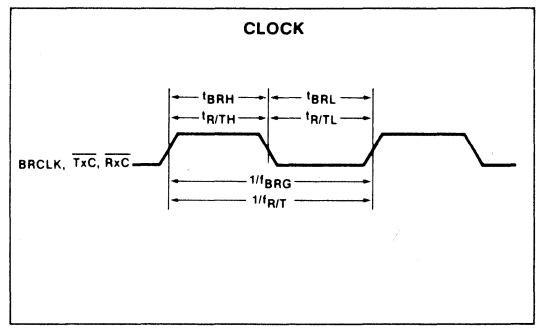
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified. See ordering code table for applicable temperature range and operating supply range.
5. All voltage measurements are referenced to ground. All time measurements are at the 50% level for inputs (except t_{BRH} and T_{BRL}) and at 0.8V and 2.0V for outputs. Input levels for testing are 0.45V and 2.4V
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. TxRDY, RxRDY and TxEMT/DSCHG outputs are open drain.
8. Parameter applies when internal transmitter clock is used.
9. Under test conditions of 5.0688MHz f_{BRG}, t_{BRH} and t_{BRL} measured at V_{IH} and V_{IL} respectively.
10. $t_{R/T}$ and $t_{R/TL}$ shown for all modes except local loopback. For local loopback mode $f_{R/T}$ = 0.7MHz and $t_{R/TL}$ = 700ns min.

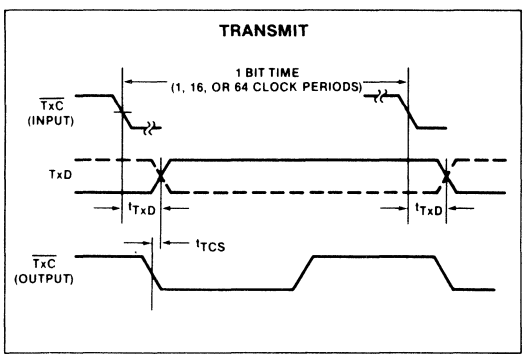
CAPACITANCE $T_A = 25°C$, $V_{CC} = 0V$

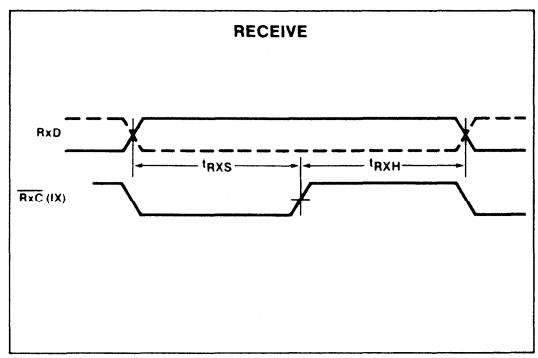
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Capacitance					pF
C_{IN}	Input	fc = 1MHz			20	
C_{OUT}	Output	Unmeasured pins tied to ground			20	
$C_{I/O}$	Input/Output				20	

TIMING DIAGRAMS



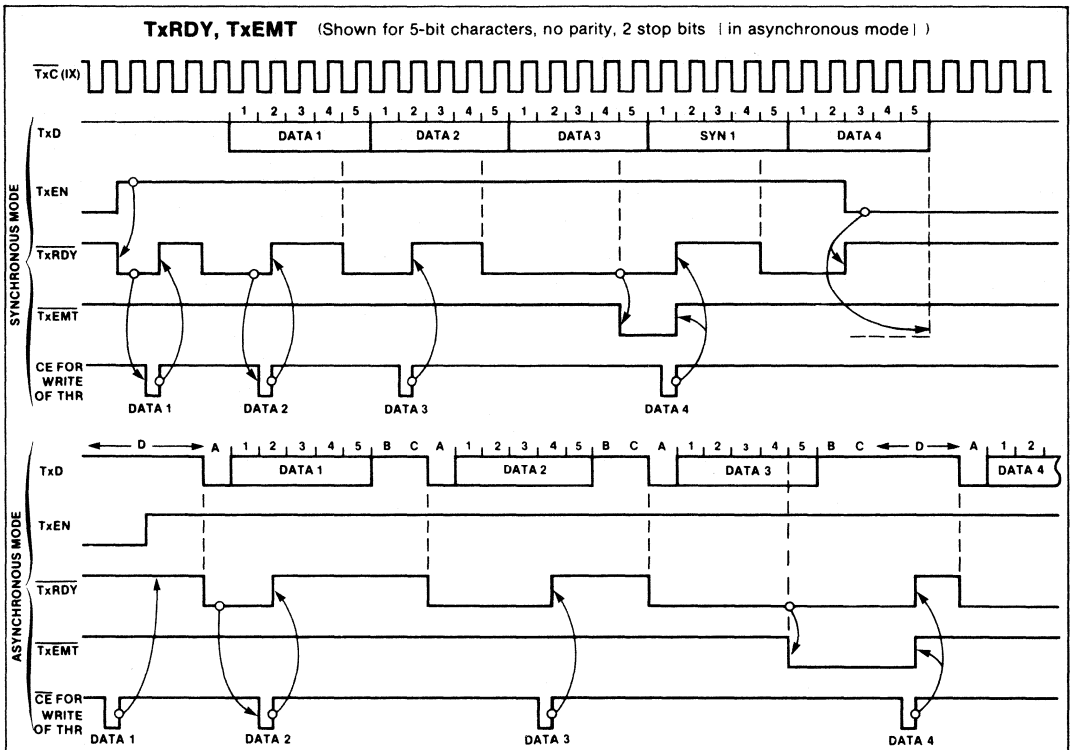






TIMING DIAGRAMS (Cont'd)

TxRDY, TxEMT (Shown for 5-bit characters, no parity, 2 stop bits [in asynchronous mode [ )

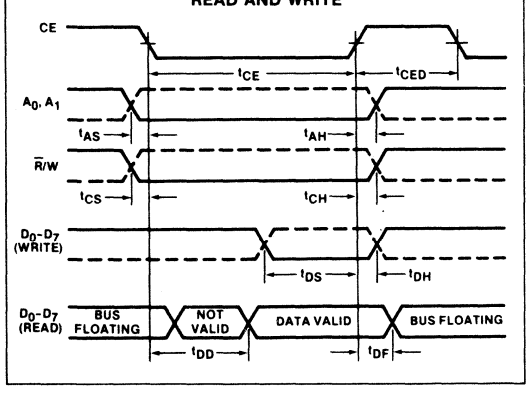


NOTES

A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition

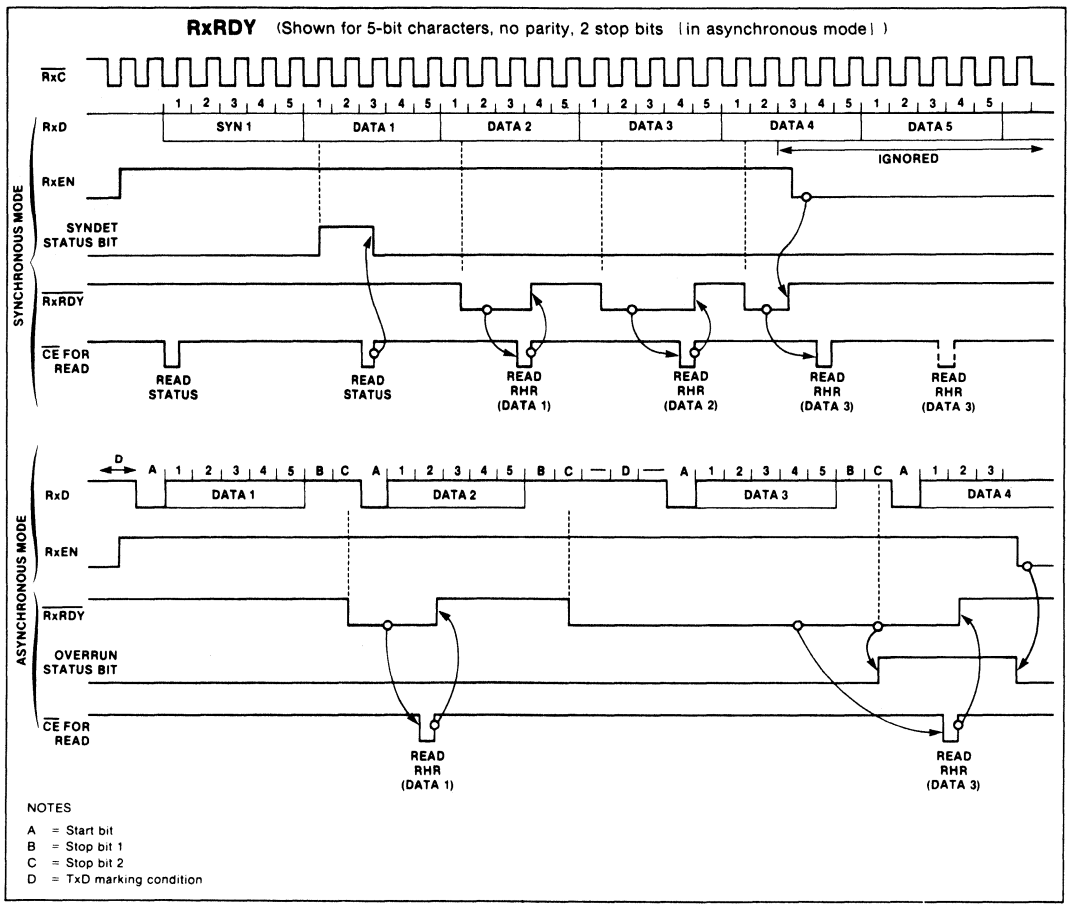
TxEMT goes low at the beginning of the last data bit, or, if parity is enabled, at the beginning of the parity bit.

READ AND WRITE



TIMING DIAGRAMS (Cont'd)

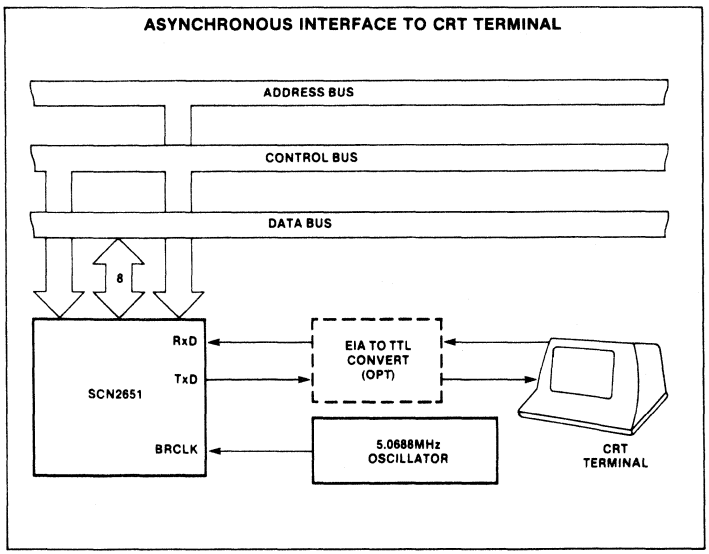
RxRDY (Shown for 5-bit characters, no parity, 2 stop bits (in asynchronous mode))

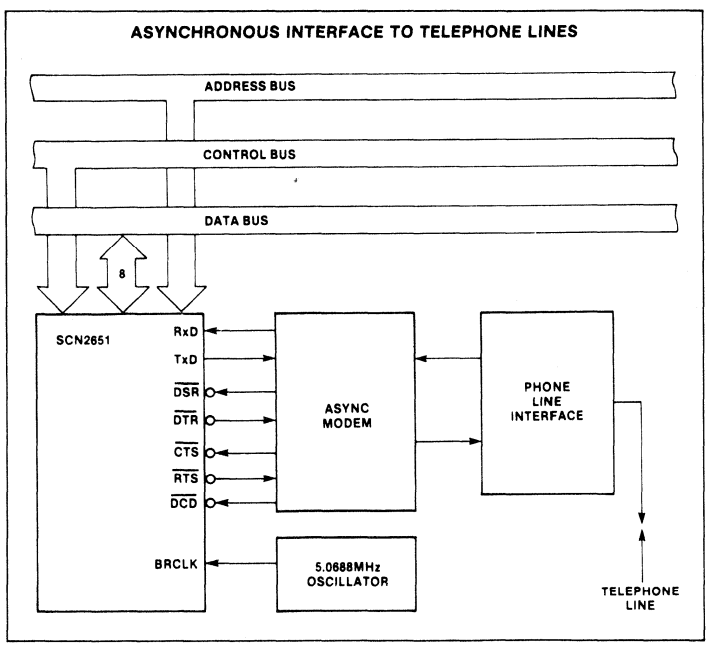


NOTES

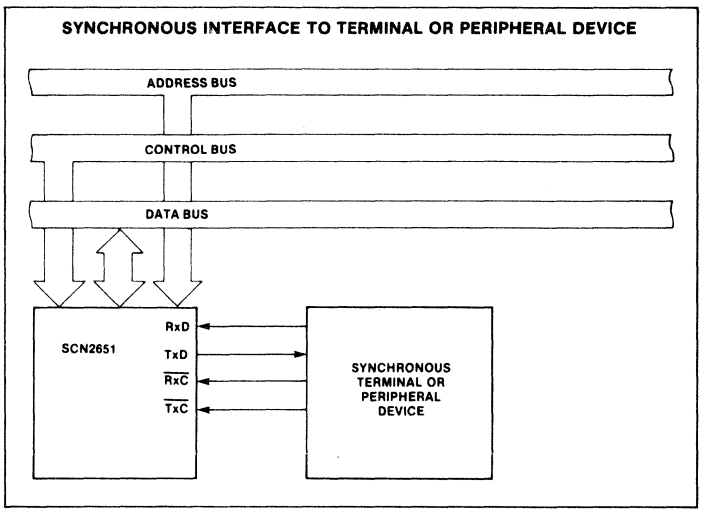
A = Start bit
B = Stop bit 1
C = Stop bit 2
D = TxD marking condition

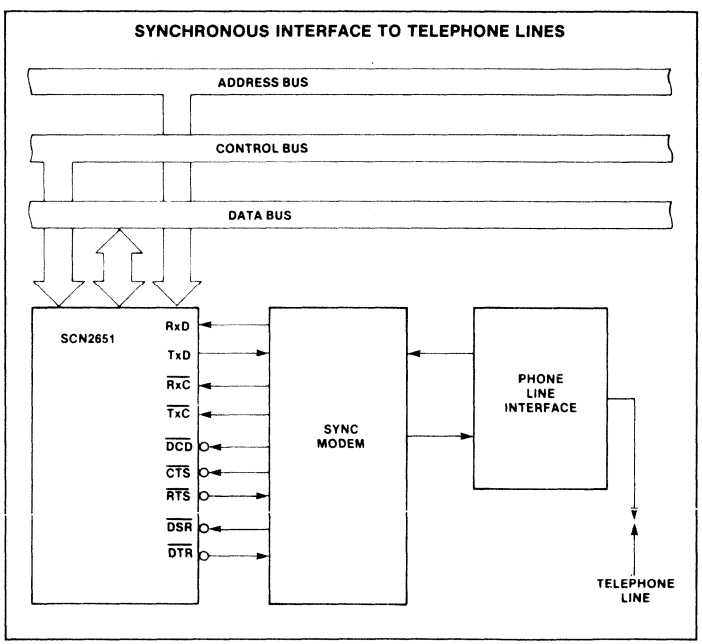
TYPICAL APPLICATIONS





TYPICAL APPLICATIONS (Cont'd)





Manufacturer reserves the right to make design and process changes and improvements.

DUAL ASYNCHRONOUS RECEIVER/TRANSMITTER (DUART)

Originally published by Signetics May 1983

Preliminary

DESCRIPTION

The Signetics SCN2681 Dual Universal Asynchronous Receiver/Transmitter (DUART) is a single chip MOS-LSI communications device that provides two independent full-duplex asynchronous receiver/transmitter channels in a single package. It interfaces directly with microprocessors and may be used in a polled or interrupt driven system.

The operating mode and data format of each channel can be programmed independently. Additionaliy, each receiver and transmitter can select its operating speed as one of eighteen fixed baud rates, a 16x clock derived from a programmable counter/timer, or an external 1x or 16x clock. The baud rate generator and counter/timer can operate directly from a crystal or from external clock inputs. The ability to independently program the operating speed of the receiver and transmitter make the DUART particularly attractive for dual-speed channel applications such as clustered terminal systems.

Each receiver is quadruply buffered to minimize the potential of receiver overrun or to reduce interrupt overhead in interrupt driven systems. In addition, a flow control capability is provided to disable a remote DUART transmitter when the buffer of the receiving device is full.

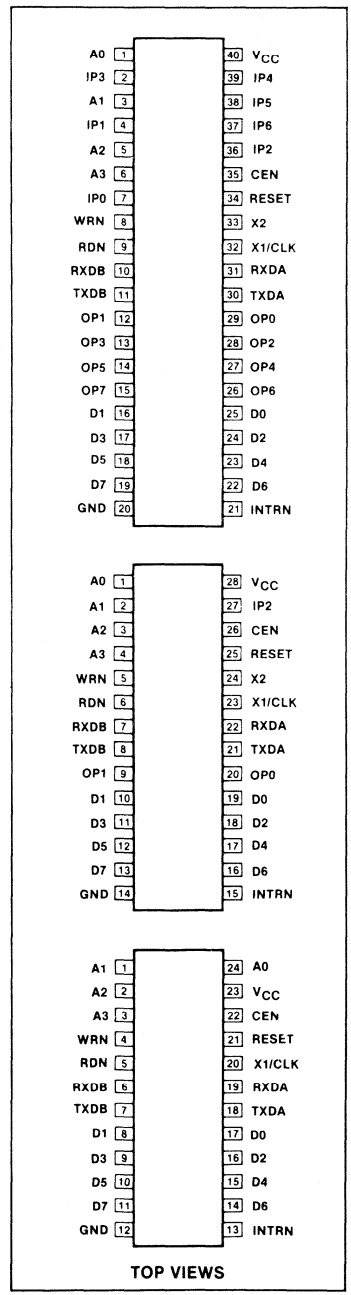
Also provided on the SCN2681 are a multipurpose 7-bit input port and a multipurpose 8-bit output port. These can be used as general purpose I/O ports or can be assigned specific functions (such as clock inputs or status/interrupt outputs) under program control.

The SCN2681 is available in three package versions to satisfy various system requirements: 40-pin and 28-pin, both 0.6″ wide DIPs, and a compact 24-pin, 0.4″ wide, DIP.

FEATURES

- **Dual full-duplex asynchronous receiver/transmiter**
- **Quadruple buffered receiver data registers**
- **Programmable data format**
  - **—5 to 8 data bits plus parity**
  - **—Odd, even, no parity or force parity**
  - **—1, 1.5 or 2 stop bits programmable in 1/16 bit increments**
- **Programmable baud rate for each receiver and transmiter selectable from:**
  - **—18 fixed rates: 50 to 38.4K baud**
  - **—One user defined rate derived from programmable timer/counter**
  - **—External 1x or 16x clock**
- **Parity, framing, and overrun error detection**
- **False start bit detection**
- **Line break detection and generation**
- **Programmable channel mode**
  - **—Normal (full duplex)**
  - **—Automatic echo**
  - **—Local loopback**
  - **—Remote loopback**
- **Multi-function programmable 16-bit counter/timer**
- **Multi-function 7-bit input port**
  - **—Can serve as clock or control inputs**
  - **—Change of state detection on four inputs**
- **Multi-function 8-bit output port**
  - **—Individual bit set/reset capability**
  - **—Outputs can be programmed to be status/interrupt signals**
- **Versatile interrupt system**
  - **—Single interrupt output with eight maskable interrupting conditions**
  - **—Output port can be configured to provide a total of up to six separate wire-OR'able interrupt outputs**
- **Maximum data transfer: 1X — 1MB/sec, 16X — 125KB/sec**
- **Automatic wake-up mode for multidrop applications**
- **Start-end break interrupt/status**
- **Detects break which originates in the middle of a character**
- **On-chip crystal oscillator**
- **TTL compatible**
- **Single +5V power supply**

PIN CONFIGURATION



TOP VIEWS

ORDERING CODE

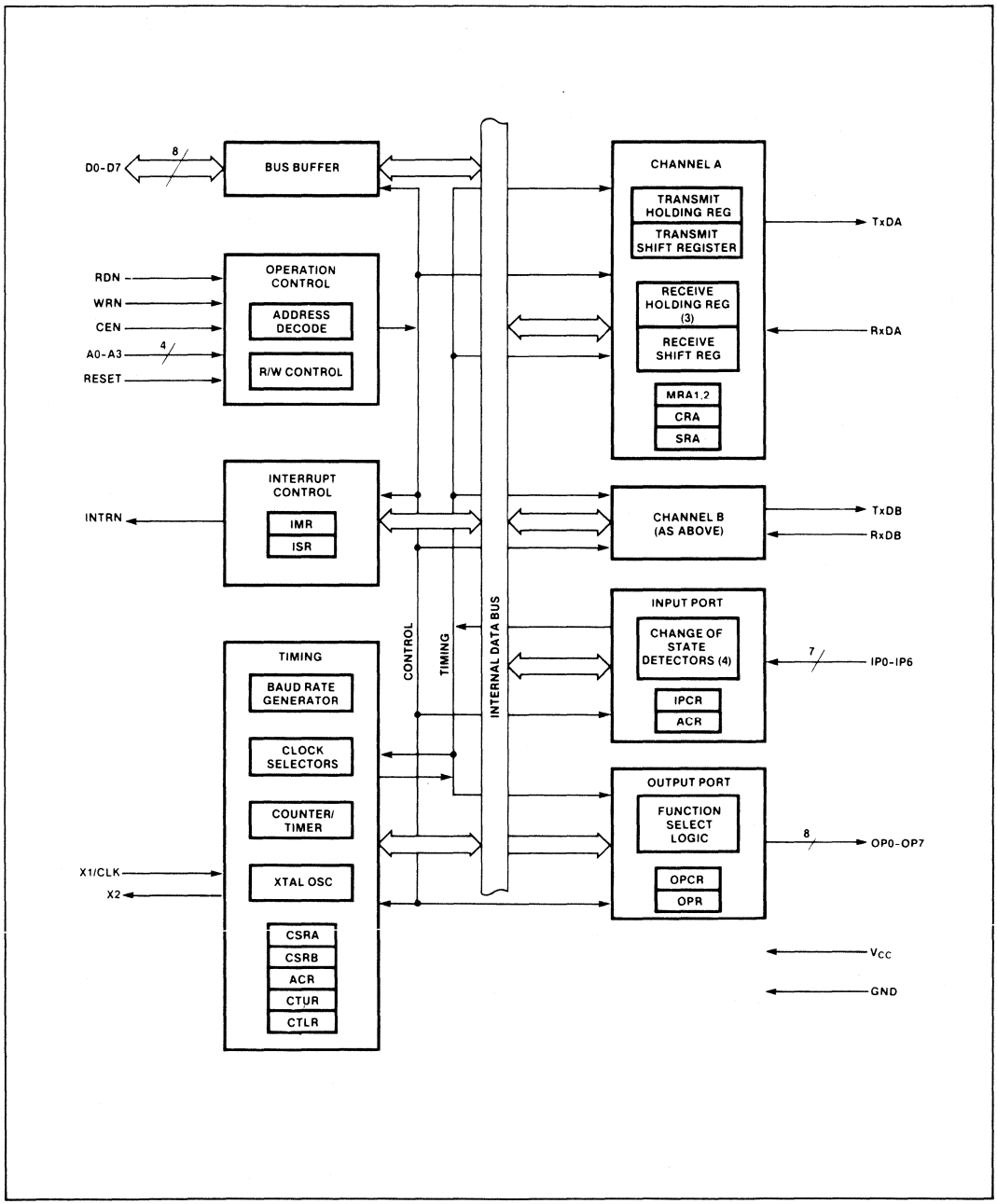
PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C		
	24 Pin[1]	28 Pin[2]	40 Pin[2]
Ceramic DIP	Not available	SCN2681AC1I28	SCN2681AC1I40
Plastic DIP	SCN2681AC1N24	SCN2681AC1N28	SCN2681AC1N40

[1] 400 mil wide DIP

[2] 600 mil wide DIP

Preliminary

BLOCK DIAGRAM



Preliminary

PIN DESIGNATION

MNEMONIC	APPLICABLE			TYPE	NAME AND FUNCTION
	40	28	24		
D0-D7	X	X	X	I/O	**Data Bus:** Bidirectional 3-state data bus used to transfer commands, data and status between the DUART and the CPU. D0 is the least significant bit.
CEN	X	X	X	I	**Chip Enable:** Active low input signal. When low, data transfers between the CPU and the DUART are enabled on D0-D7 as controlled by the WRN, RDN and A0-A3 inputs. When high, places the D0-D7 lines in the 3-state condition.
WRN	X	X	X	I	**Write Strobe:** When low and CEN is also low, the contents of the data bus is loaded into the addressed register. The transfer occurs on the rising edge of the signal.
RDN	X	X	X	I	**Read Strobe:** When low and CEN is also low, causes the contents of the addressed register to be presented on the data bus. The read cycle begins on the falling edge of RDN.
A0-A3	X	X	X	I	**Address Inputs:** Select the DUART internal registers and ports for read/write operations.
RESET	X	X	X	I	**Reset:** A high level clears internal registers (SRA, SRB, IMR, ISR, OPR, OPCR), puts OP0-OP7 in the high state, stops the counter/timer, and puts channels A and B in the inactive state, with the TxDA and TxDB outputs in the mark (high) state.
INTRN	X	X	X	O	**Interrupt Request:** Active low, open drain, output which signals the CPU that one or more of the eight maskable interrupting conditions are true.
X1/CLK	X	X	X	I	**Crystal 1:** Crystal or external clock input. A crystal or clock of the specified limits must be supplied at all times. When a crystal is used, a capacitor must be connected from this pin to ground (see figure 5).
X2	X	X		O	**Crystal 2:** Connection for other side of the crystal. Should be connected to ground if a crystal is not used. When a crystal is used, a capacitor must be connected from this pin to ground (see figure 5).
RxDA	X	X	X	I	**Channel A Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
RxDB	X	X	X	I	**Channel B Receiver Serial Data Input:** The least significant bit is received first. 'Mark' is high, 'space' is low.
TxDA	X	X	X	O	**Channel A Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
TxDB	X	X	X	O	**Channel B Transmitter Serial Data Output:** The least significant bit is transmitted first. This output is held in the 'mark' condition when the transmitter is disabled, idle, or when operating in local loopback mode. 'Mark' is high, 'space' is low.
OP0	X	X		O	**Output 0:** General purpose output, or channel A request to send (RTSAN, active low). Can be deactivated on receive or transmit.
OP1	X	X		O	**Output 1:** General purpose output, or channel B request to send (RTSBN, active low). Can be deactivated on receive or transmit.
OP2	X			O	**Output 2:** General purpose output, or channel A transmitter 1X or 16X clock output, or channel A receiver 1X clock output.
OP3	X			O	**Output 3:** General purpose output, or open drain, active low counter/timer output, or channel B transmitter 1X clock output, or channel B receiver 1X clock output.
OP4	X			O	**Output 4:** General purpose output, or channel A open drain, active low, RxRDYA/FFULLA output.
OP5	X			O	**Output 5:** General purpose output, or channel B open drain, active low, RxRDYB/FFULLB output.
OP6	X			O	**Output 6:** General purpose output, or channel A open drain, active low, TxRDYA output.
OP7	X			O	**Output 7:** General purpose output, or channel B open drain, active low, TxRDYB output.
IP0	X			I	**Input 0:** General purpose input, or channel A clear to send active low input (CTSAN).
IP1	X			I	**Input 1:** General purpose input, or channel B clear to send active low input (CTSBN).
IP2	X	X		I	**Input 2:** General purpose input, or counter/timer external clock input.
IP3	X			I	**Input 3:** General purpose input, or channel A transmitter external clock input (TxCA). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.

Preliminary

PIN DESIGNATION (Continued)

MNEMONIC	APPLICABLE			TYPE	NAME AND FUNCTION
	40	28	24		
IP4	X			I	**Input 4:** General purpose input, or channel A receiver external clock input (RxCA). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
IP5	X			I	**Input 5:** General purpose input, or channel B transmitter external clock input (TxCB). When the external clock is used by the transmitter, the transmitted data is clocked on the falling edge of the clock.
IP6	X			I	**Input 6:** General purpose input or channel B receiver external clock input (RxCB). When the external clock is used by the receiver, the received data is sampled on the rising edge of the clock.
V_{CC}	X	X	X	I	**Power Supply:** + 5V supply input
GND	X	X	X	I	**Ground**

BLOCK DIAGRAM

The 2681 DUART consists of the following eight major sections: data bus buffer, operation control, interrupt control, timing, communications channels A and B, input port and output port. Refer to the block diagram.

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the DUART.

Operation Control

The operation control logic receives operation commands from the CPU and generates appropriate signals to internal sections to control device operation. It contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer.

Interrupt Control

A single active low interrupt output (INTRN) is provided which is activated upon the occurence of any of eight internal events. Associated with the interrupt system are the interrupt mask register (IMR) and the interrupt status register (ISR). The IMR may be programmed to select only certain conditions to cause INTRN to be asserted. The ISR can be read by the CPU to determine all currently active interrupting conditions.

Outputs OP3-OP7 can be programmed to provide discrete interrupt outputs for the transmitters, receivers, and counter/timer.

Timing Circuits

The timing block consists of a crystal oscillator, a baud rate generator, a programmable 16-bit counter/timer, and four clock selectors. The crystal oscillator operates directly from a 3.6864MHz crystal connected across the X1/CLK and X2 inputs. If an external clock of the appropriate frequency is available, it may be connected to X1/CLK. The clock serves as the basic timing reference for the baud rate generator (BRG), the counter/timer, and other internal circuits. A clock signal within the limits specified in the specifications section of this data sheet must always be supplied to the DUART.

The baud rate generator operates from the oscillator or external clock input and is capable of generating 18 commonly used data communications baud rates ranging from 50 to 38.4K baud. The clock outputs from the BRG are at 16X the actual baud rate. The counter/timer can be used as a timer to produce a 16X clock for any other baud rate by counting down the crystal clock or an external clock. The four clock selectors allow the independent selection, for each receiver and transmitter, of any of these baud rates or an external timing signal.

The counter/timer (C/T) can be programmed to use one of several timing sources as its input. The output of the C/T is available to the clock selectors and can also be programmed to be output at OP3. In the counter mode, the contents of the C/T can be read by the CPU and it can be stopped and started under program control. In the timer mode, the C/T acts as a programmable divider.

Communications Channels A and B

Each communications channel of the 2681 comprises a full duplex asynchronous receiver/transmitter (UART). The operating frequency for each receiver and transmitter can be selected independently from the baud rate generator, the counter timer, or from an external input.

The transmitter accepts parallel data from the CPU, converts it to a serial bit stream, inserts the appropriate start, stop, and optional parity bits and outputs a composite serial stream of data on the TxD output pin. The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition and sends an assembled character to the CPU.

Input Port

The inputs to this unlatched 7-bit port can be read by the CPU by performing a read operation at address D_{16}. A high input results in a logic 1 while a low input results in a logic 0. D_7 will always be read as a logic 1. The pins of this port can also serve as auxiliary inputs to certain portions of the DUART logic.

Four change-of-state detectors are provided which are associated with inputs IP3, IP2, IP1, and IP0. A high-to-low or low-to-high transition of these inputs lasting longer than 25-50μs will set the corresponding bit in the input port will change register. The bits are cleared when the register is read by the CPU. Any change of state can also be programmed to generate an interrupt to the CPU.

Preliminary

Output Port

The 8-bit multi-purpose output port can be used as a general purpose output port, in which case the outputs are the complements of the output port register (OPR). OPR[n] = 1 results in OP[n] = low and vice-versa. Bits of the OPR can be individually set and reset. A bit is set by performing a write operation at address E_{16} with the accompanying data specifying the bits to be set (1 = set, 0 = no change). Likewise, a bit is reset by a write at address F_{16} with the accompanying data specifying the bits to be reset (1 = reset, 0 = no change).

Outputs can be also individually assigned specific functions by appropriate programming of the channel A mode registers (MR1A, MR2A), the channel B mode registers (MR1B, MR2B), and the output port configuration register (OPCR).

OPERATION

Transmitter

The 2681 is conditioned to transmit data when the transmitter is enabled through the command register. The 2681 indicates to the CPU that it is ready to accept a character by setting the TxRDY bit in the status register. This condition can be programmed to generate an interrupt request at OP6 or OP7 and INTRN. When a character is loaded into the transmit holding register (THR), the above conditions are negated. Data is transferred from the holding register to the transmit shift register when it is idle or has completed transmission of the previous character. The TxRDY conditions are then asserted again which means one full character time of buffering is provided. Characters cannot be loaded into the THR while the transmitter is disabled.

The transmitter converts the parallel data from the CPU to a serial bit stream on the TxD output pin. It automatically sends a start bit followed by the programmed number of data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the THR, the TxD output remains high and the TxEMT bit in the status register (SR) will be set to 1. Transmission resumes and the TxEMT bit is cleared when the CPU loads a new character into the THR. If the transmitter is disabled, it continues operating until the character currently being transmitted is completely sent out. The transmitter can be forced to send a continuous low condition by issuing a send break command.

The transmitter can be reset through a software command. If it is reset, operation ceases immediately and the transmitter must be enabled through the command register before resuming operation. If CTS operation is enabled, the CTSN input must be low in order for the character to be transmitted. if it goes high in the middle of a transmission, the character in the shift register is transmitted and TxDA then remains in the marking state until CTSN goes low. The transmitter can also control the deactivation of the RTSN output. If programmed, the RTSN output will be reset one bit time after the character in the transmit shift register and transmit holding register (if any) are completely transmitted, if the transmitter has been disabled.

Receiver

The 2681 is conditioned to receive data when enabled through the command register. The receiver looks for a high to low (mark to space) transition of the start bit on the RxD input pin. If a transition is detected, the state of the RxD pin is sampled each 16X clock for 7-1/2 clocks (16X clock mode) or at the next rising edge of the bit time clock (1X clock mode). If RxD is sampled high, the start bit is invalid and the search for a valid start bit begins again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least sigificant bit is received first. The data is then transferred to the receive holding register (RHR) and the RxRDY bit in the SR is set to a 1. This condition can be programmed to generate an interrupt at OP4 or OP5 and INTRN. If the character length is less than eight bits, the most significant unused bits in the RHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (framing error) and RxD remains low for one half of the bit period after the stop bit was sampled, then the receiver operates as if a new start bit transition had been detected at that point (one-half bit time after the stop bit was sampled).

The parity error, framing error, overrun error and received break state (if any) are strobed into the SR at the received character boundary, before the RxRDY status bit is set. If a break condition is detected (RxD is low for the entire character including the stop bit), a character consisting of all zeros will be loaded into the RHR and the received break bit in the SR is set to 1. The RxD input must return to a high condition for at least one-half bit time before a search for the next start bit begins.

The RHR consists of a first-in-first-out (FIFO) stack with a capacity of three characters. Data is loaded from the receive shift register into the topmost empty position of the FIFO. The RxRDY bit in the status register is set whenever one or more characters are available to be read, and a FFULL status bit is set if all three stack positions are filled with data. Either of these bits can be selected to cause an interrupt. A read of the RHR outputs the data at the top of the FIFO. After the read cycle, the data FIFO and its associated status bits (see below) are 'popped' thus emptying a FIFO position for new data.

In addition to the data word, three status bits (parity error, framing error, and received break) are also appended to each data character in the FIFO (overrun is not). Status can be provided in two ways, as programmed by the error mode control bit in the mode register. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these three bits is the logical OR of the status for all characters coming to the top of the FIFO since the last 'reset error' command was issued. In either mode reading the SR does not affect the FIFO. The FIFO is 'popped' only when the RHR is read. Therefore the status register should be read prior to reading the FIFO.

If the FIFO is full when a new character is received, that character is held in the receive shift register until a FIFO position is available. If an additional character is received while this state exits, the contents of the FIFO are not affected: the character previously in the shift register is lost and the overrun error status bit (SR[4]) will be set upon receipt of the start bit of the new (overruning) character.

The receiver can control the deactivation of RTS. If programmed to operate in this mode, the RTSN output will be negated when a valid start bit was received and the FIFO is full. When a FIFO position becomes available, the RTSN output will be re-asserted automatically. This feature can be used to prevent an overrun, in the

Preliminary

receiver, by connecting the RTSN output to the CTSN input of the transmitting device.

If the receiver is disabled, the FIFO characters can be read. However, no additional characters can be received until the receiver is enabled again. If the receiver is reset, the FIFO and all of the receiver status, and the corresponding output ports and interrupt are reset. No additional characters can be received until the receiver is enabled again.

Multidrop Mode

The DUART is equipped with a wake up mode used for multidrop applications. This mode is selected by programming bits MR1A[4:3] or MR1B[4:3] to '11' for channels A and B respectively. In this mode of operation, a 'master' station transmits an address character followed by data characters for the addressed 'slave' station. The slave stations, with receivers that are normally disabled, examine the received data stream and 'wake-up' the CPU (by setting RxRDY) only upon receipt of an address character. The CPU compares the received address to its station address and enables the receiver if it wishes to receive the subsequent data characters. Upon receipt of another address character, the CPU may disable the receiver to initiate the process again.

A transmitted character consists of a start bit, the programmed number of data bits, an address/data (A/D) bit, and the programmed number of stop bits. The polarity of the transmitted A/D bit is selected by the CPU by programming bit MR1A[2]/MR1B[2]. MR1A[2]/MR1B[2] = 0 transmits a zero in the A/D bit position, which identifies the corresponding data bits as data, while MR1A[2]/MR1B[2] = 1 transmits a one in the A/D bit position, which identifies the corresponding data bits as an address. The CPU should program the mode register prior to loading the corresponding data bits into the THR.

In this mode, the receiver continuously looks at the received data stream, whether it is enabled or disabled. If disabled, it sets the RxRDY status bit and loads the character into the RHR FIFO if the received A/D bit is a one (address tag), but discards the received character if the received A/D bit is a zero (data tag). If enabled, all received characters are transferred to the CPU via the RHR. In either case, the data bits are loaded into the data FIFO while the A/D bit is loaded into the status FIFO position normally used for parity error (SRA[5] or SRB[5]). Framing error, overrun error, and break detect operate normally whether or not the receiver is enabled.

PROGRAMMING

The operation of the DUART is programmed by writing control words into the appropriate registers. Operational feedback is provided via status registers which can be read by the CPU. The addressing of the registers is described in table 1.

The contents of certain control registers are initialized to zero on RESET. Care should be exercised if the contents of a register are changed during operation, since certain changes may cause operational problems. For example, changing the number of bits per character while the transmitter is active may cause the transmission of an incorrect character. In general, the contents of the MR, the CSR, and the OPCR should only be changed while the receiver(s) and transmitter(s) are not enabled, and certain changes to the ACR should only be made while the C/T is stopped.

Mode registers 1 and 2 of each channel are accessed via independent auxiliary pointers. The pointer is set to MR1x by RESET or by issuing a 'reset pointer' command via the corresponding command register. Any read or write of the mode register while the pointer is at MR1x switches the pointer to MR2x. The pointer then remains at MR2x, so that subsequent accesses are always to MR2x unless the pointer is reset to MR1x as described above.

Mode, command, clock select, and status registers are duplicated for each channel to provide total independent operation and control. Refer to table 2 for register bit descriptions.

MR1A — Channel A Mode Register 1

MR1A is accessed when the channel A MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRA. After reading or writing MR1A, the pointer will point to MR2A.

MR1A[7] — Channel A Receiver Request-to-Send Control — This bit controls the deactivation of the RTSAN output (OP0) by the receiver. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR1A[7] = 1 causes RTSAN to be negated upon receipt of a valid start bit if the channel A FIFO is full. However, OPR[0] is not reset and RTSAN will be asserted again when an empty FIFO position is available. This feature can be used for flow control to prevent overrun in the receiver by using the RTSAN output signal to control the CTSN input of the transmitting device.

MR1A[6] — Channel A Receiver Interrupt Select — This bit selects either the channel A receiver ready status (RXRDY) or the channel A FIFO full status (FFULL) to be used for CPU interrupts. It also causes the selected bit to be output on OP4 if it is programmed as an interrupt output via the OPCR.

MR1A[5] — Channel A Error Mode Select — This bit selects the operating mode of the three FIFOed status bits (FE, PE, received break) for channel A. In the 'character' mode, status is provided on a character-by-character basis: the status applies only to the character at the top of the FIFO. In the 'block' mode, the status provided in the SR for these bits is the ac-

Table 1 2681 REGISTER ADDRESSING

A3	A2	A1	A0	READ (RDN = 0)	WRITE (WRN = 0)
0	0	0	0	Mode Register A (MR1A, MR2A)	Mode Register A (MR1A, MR2A)
0	0	0	1	Status Register A (SRA)	Clock Select Reg. A (CSRA)
0	0	1	0	*Reserved*	Command Register A (CRA)
0	0	1	1	RX Holding Register A (RHRA)	TX Holding Register A (THRA)
0	1	0	0	Input Port Change Reg. (IPCR)	Aux. Control Register (ACR)
0	1	0	1	Interrupt Status Reg. (ISR)	Interrupt Mask Reg. (IMR)
0	1	1	0	Counter/Timer Upper (CTU)	C/T Upper Register (CTUR)
0	1	1	1	Counter/Timer Lower (CTL)	C/T Lower Register (CTLR)
1	0	0	0	Mode Register B (MR1B, MR2B)	Mode Register B (MR1B, MR2B)
1	0	0	1	Status Register B (SRB)	Clock Select Reg. B (CSRB)
1	0	1	0	*Reserved*	Command Register B (CRB)
1	0	1	1	RX Holding Register B (RHRB)	TX Holding Register B (THRB)
1	1	0	0	*Reserved*	*Reserved*
1	1	0	1	Input Port	Output Port Conf. Reg. (OPCR)
1	1	1	0	Start Counter Command	Set Output Port Bits Command
1	1	1	1	Stop Counter Command	Reset Output Port Bits Command

Preliminary

Table 2 REGISTER BIT FORMATS

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RX RTS CONTROL	RX INT SELECT	ERROR MODE	PARITY MODE		PARITY TYPE	BITS PER CHAR.	
MR1A MR1B	0 = no 1 = yes	0 = RXRDY 1 = FFULL	0 = char 1 = block	00 = with parity 01 = force parity 10 = no parity 11 = multi-drop mode		0 = even 1 = odd	00 = 5 01 = 6 10 = 7 11 = 8	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	CHANNEL MODE		Tx RTS CONTROL	CTS ENABLE Tx	STOP BIT LENGTH*			
MR2A MR2B	00 = Normal 01 = Auto echo 10 = Local loop 11 = Remote loop		0 = no 1 = yes	0 = no 1 = yes	0 = 0.563 1 = 0.625 2 = 0.688 3 = 0.750	4 = 0.813 5 = 0.875 6 = 0.938 7 = 1.000	8 = 1.563 9 = 1.625 A = 1.688 B = 1.750	C = 1.813 D = 1.875 E = 1.938 F = 2.000

*Add 0.5 to values shown for 0-7 if channel is programmed for 5 bits/char.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CSRA	RECEIVER CLOCK SELECT				TRANSMITTER CLOCK SELECT			
CSRB	See text				See text			

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	not used—must be 0	MISCELLANEOUS COMMANDS			DISABLE Tx	ENABLE Tx	DISABLE Rx	ENABLE Rx
CRA CRB		See text			0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	RECEIVED BREAK	FRAMING ERROR	PARITY ERROR	OVERRUN ERROR	TxEMT	TxRDY	FFULL	RxRDY
SRA SRB	0 = no 1 = yes *	0 = no 1 = yes *	0 = no 1 = yes *	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

*These status bits are appended to the corresponding data character in the receive FIFO. A read of the status register provides these bits (7:5) from the top of the FIFO together with bits 4:0. These bits are cleared by a 'reset error status' command. In character mode they are discarded when the corresponding data character is read from the FIFO.

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	OP7	OP6	OP5	OP4	OP3		OP2	
OPCR	0 = OPR[7] 1 = TxRDYB	0 = OPR[6] 1 = TxRDYA	0 = OPR[5] 1 = RxRDY/ FFULLB	0 = OPR[4] 1 = RxRDY/ FFULLA	00 = OPR[3] 01 = C/T OUTPUT 10 = TxCB (1X) 11 = RxCB (1X)		00 = OPR[2] 01 = TxCA (16X) 10 = TxCA (1X) 11 = RxCA (1X)	

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	BRG SET SELECT	COUNTER/TIMER MODE AND SOURCE			DELTA IP3 INT	DELTA IP2 INT	DELTA IP1 INT	DELTA IP0 INT
ACR	0 = set1 1 = set2	See table 4			0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	DELTA IP3	DELTA IP2	DELTA IP1	DELTA IP0	IP3	IP2	IP1	IP0
IPCR	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high	0 = low 1 = high

Preliminary

Table 2 REGISTER BIT FORMATS (Continued)

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ISR	INPUT PORT CHANGE	DELTA BREAK B	RxRDY/ FFULLB	TxRDYB	COUNTER READY	DELTA BREAK A	RxRDY/ FFULLA	TxRDYA
	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes	0 = no 1 = yes

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
IMR	IN. PORT CHANGE INT	DELTA BREAK B INT	RxRDY/ FFULLB INT	TxRDYB INT	COUNTER READY INT	DELTA BREAK A INT	RxRDY/ FFULLA INT	TxRDYA INT
	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on	0 = off 1 = on

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CTUR	C/T[15]	C/T[14]	C/T[13]	C/T[12]	C/T[11]	C/T[10]	C/T[9]	C/T[8]

	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CTLR	C/T[7]	C/T[6]	C/T[5]	C/T[4]	C/T[3]	C/T[2]	C/T[1]	C/T[0]

cumulation (logical OR) of the status for all characters coming to the top of the FIFO since the last 'reset error' command for channel A was issued.

MR1A[4:3] — Channel A Parity Mode Select — If 'with parity' or 'force parity' is selected, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. MR1A[4:3] = 11 selects channel A to operate in the special multidrop mode described in the Operation section.

MR1A[2] — Channel A Parity Type Select — This bit selects the parity type (odd or even) if the 'with parity' mode is programmed by MR1A[4:3], and the polarity of the forced parity bit if the 'force parity' mode is programmed. It has no effect if the 'no parity' mode is programmed. In the special multidrop mode it selects the polarity of the A/D bit.

MR1A[1:0] — Channel A Bits per Character Select — This field selects the number of data bits per character to be transmitted and received. The character length does not include the start, parity, and stop bits.

MR2A — Channel A Mode Register 2

MR2A is accessed when the channel A MR pointer points to MR2, which occurs after any access to MR1A. Accesses to MR2A do not change the pointer.

MR2A[7:6] — Channel A Mode Select — Each channel of the DUART can operate in one of four modes. MR2A[7:6] = 00 is the normal mode, with the transmitter and receiver operating independently. MR2A[7:6] = 01 places the channel in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Received data is reclocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. The channel A TxRDY and TxEMT status bits are inactive.
5. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Character framing is checked, but the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.
8. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.

Two diagnostic modes can also be configured. MR2A[7:6] = 10 selects local loopback mode. In this mode:

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TxDA output is held high.
4. The RxDA input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode, selected by MR2A[7:6] = 11. In this mode:

1. Received data is relocked and retransmitted on the TxDA output.
2. The receive clock is used for the transmitter.

Preliminary

3. Received data is not sent to the local CPU, and the error status conditions are inactive.
4. The received parity is not checked and is not regenerated for transmission, i.e., transmitted parity bit is as received.
5. The receiver must be enabled.
6. Character framing is not checked, and the stop bits are retransmitted as received.
7. A received break is echoed as received until the next valid start bit is detected.

The user must exercise care when switching into and out of the various modes. The selected mode will be activated immediately upon mode selection, even if this occurs in the middle of a received or transmitted character. Likewise, if a mode is deselected, the device will switch out of the mode immediately. An exception to this is switching out of autoecho or remote loopback modes: if the de-selection occurs just after the receiver has sampled the stop bit (indicated in autoecho by assertion of RxRDY), and the transmitter is enabled, the transmitter will remain in autoecho mode until the entire stop bit has been retransmitted.

MR2A[5] — Channel A Transmitter Request-to-Send Control — This bit controls the deactivation of the RTSAN output (OP0) by the transmitter. This output is normally asserted by setting OPR[0] and negated by resetting OPR[0]. MR2A[5] = 1 causes OPR[0] to be reset automatically one bit time after the characters in the channel A transmit shift register and in the THR, if any, are completely transmitted, including the programmed number of stop bits, if the transmitter is not enabled. This feature can be used to automatically terminate the transmission of a message as follows:

1. Program auto-reset mode: MR2A[5] = 1.
2. Enable transmitter.
3. Assert RTSAN: OPR[0] = 1.
4. Send message.
5. Disable transmitter after the last character is loaded into the channel A THR.
6. The last character will be transmitted and OPR[0] will be reset one bit time after the last stop bit, causing RTSAN to be negated.

MR2A[4] — Channel A Clear-to-Send Control — If this bit is 0, CTSAN has no effect on the transmitter. If this bit is a 1, the transmitter checks the state of CTSAN (IP0) each time it is ready to send a character. If IP0 is asserted (low), the character is transmitted. If it is negated (high), the TxDA output remains in the marking state and the transmission is delayed until CTSAN goes low. Changes in CTSAN while a character is being transmitted do not affect the transmission of that character.

MR2A[3:0] — Channel A Stop Bit Length Select — This field programs the length of the stop bit appended to the transmitted character. Stop bit lengths of 9/16 to 1 and 1-9/16 to 2 bits, in increments of 1/16 bit, can be programmed for character lengths of 6, 7, and 8 bits. For a character length of 5 bits, 1-1/16 to 2 stop bits can be programmed in increments of 1/16 bit. The receiver only checks for a 'mark' condition at the center of the first stop bit position (one bit time after the last data bit, or after the parity bit if parity is enabled) in all cases.

If an external 1X clock is used for the transmitter, MR2A[3] = 0 selects one stop bit and MR2A[3] = 1 selects two stop bits to be transmitted.

MR1B — Channel B Mode Register 1

MR1B is accessed when the channel B MR pointer points to MR1. The pointer is set to MR1 by RESET or by a 'set pointer' command applied via CRB. After reading or writing MR1B, the pointer will point to MR2B.

The bit definitions for this register are identical to the bit definitions for MR1A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

MR2B — Channel B Mode Register 2

MR2B is accessed when the channel B MR pointer points to MR2, which occurs after any access to MR1B. Accesses to MR2B do not change the pointer.

The bit definitions for this register are identical to the bit definitions for MR2A, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

CSRA — Channel A Clock Select Register

CSRA[7:4] — Channel A Receiver Clock Select — This field selects the baud rate clock for the channel A receiver as follows:

				Baud Rate CLOCK = 3.6864MHz	
CSRA[7:4]				ACR[7] = 0	ACR[7] = 1
0	0	0	0	50	75
0	0	0	1	110	110
0	0	1	0	134.5	134.5
0	0	1	1	200	150
0	1	0	0	300	300
0	1	0	1	600	600
0	1	1	0	1,200	1,200
0	1	1	1	1,050	2,000
1	0	0	0	2,400	2,400
1	0	0	1	4,800	4,800
1	0	1	0	7,200	1,800
1	0	1	1	9,600	9,600
1	1	0	0	38.4K	19.2K
1	1	0	1	Timer	Timer
1	1	1	0	IP4—16X	IP4—16X
1	1	1	1	IP4—1X	IP4—1X

The receiver clock is always a 16X clock except for CSRA[7:4] = 1111.

CSRA[3:0] — Channel A Transmitter Clock Select — This field selects the baud rate clock for the channel A transmitter. The field definition is as per CSRA[7:4] except as follows:

				Baud Rate	
CSRA[3:0]				ACR[7] = 0	ACR[7] = 1
1	1	1	0	IP3—16X	IP3—16X
1	1	1	1	IP3—1X	IP3—1X

The transmitter clock is always a 16X clock except for CSRA[3:0] = 1111.

CSRB — Channel B Clock Select Register

CSRB[7:4] — Channel B Receiver Clock Select — This field selects the baud rate clock for the channel B receiver. The field definition is as per CSRA[7:4] except as follows:

				Baud Rate	
CSRB[7:4]				ACR[7] = 0	ACR[7] = 1
1	1	1	0	IP6—16X	IP6—16X
1	1	1	1	IP6—1X	IP6—1X

The receiver clock is always a 16X clock except for CSRB[7:4] = 1111.

CSRB[3:0] — Channel B Transmitter Clock Select — This field selects the baud rate clock for the channel B transmitter. The field definition is as per CSRA[7:4] except as follows:

				Baud Rate	
CSRB[3:0]				ACR[7] = 0	ACR[7] = 1
1	1	1	0	IP5—16X	IP5—16X
1	1	1	1	IP5—1X	IP5—1X

The transmitter clock is always a 16X clock except for CSRB[3:0] = 1111.

Preliminary

CRA — Channel A Command Register

CRA is a register used to supply commands to channel A. Multiple commands can be specified in a single write to CRA as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

CRA[6:4] — Channel A Miscellaneous Commands — The encoded value of this field may be used to specify a single command as follows:

CRA[6:4]	COMMAND
0 0 0	No command.
0 0 1	Reset MR pointer. Causes the channel A MR pointer to point to MR1.
0 1 0	Reset receiver. Resets the channel A receiver as if a hardware reset had been applied. The receiver is disabled and the FIFO is flushed.
0 1 1	Reset transmitter. Resets the channel A transmitter as if a hardware reset had been applied.
1 0 0	Reset error status. Clears the channel A Received Break, Parity Error, Framing Error, and Overrun Error bits in the status register (SRA[7:4]). Used in character mode to clear OE status (although RB, PE, and FE bits will also be cleared) and in block mode to clear all error status after a block of data has been received.
1 0 1	Reset channel A break change interrupt. Causes the channel A break detect change bit in the interrupt status register (ISR[2]) to be cleared to zero.
1 1 0	Start break. Forces the TXDA output low (spacing). If the transmitter is empty the start of the break condition will be delayed up to two bit times. If the transmitter is active the break begins when transmission of the character is completed. If a character is in the THR, the start of the break will be delayed until that character, or any others loaded subsequently are transmitted. The transmitter must be enabled for this command to be accepted.
1 1 1	Stop Break. The TXDA line will go high (marking) within two bit times. TXDA will remain high for one bit time before the next character, if any, is transmitted.

CRA[3] — Disable Channel A Transmitter — This command terminates transmitter operation and resets the TxRDY and TxEMT status bits. However, if a character is being transmitted or if a character is in the THR when the transmitter is disabled, the transmission of the character(s) is completed before assuming the inactive state.

CRA[2] — Enable Channel A Transmitter — Enables operation of the channel A transmitter. The TxRDY status bit will be asserted.

CRA[1] — Disable Channel A Receiver — This command terminates operation of the receiver immediately — a character being received will be lost. The command has no effect on the receiver status bits or any other control registers. If the special multidrop mode is programmed, the receiver operates even if it is disabled. See Operation section.

CRA[0] — Enable Channel A Receiver — Enables operation of the channel A receiver. If not in the special wakeup mode, this also forces the receiver into the search for start-bit state.

CRB — Channel B Command Register

CRB is a register used to supply commands to channel B. Multiple commands can be specified in a single write to CRB as long as the commands are non-conflicting, e.g., the 'enable transmitter' and 'reset transmitter' commands cannot be specified in a single command word.

The bit definitions for this register are identical to the bit definitions for CRA, except that all control actions apply to the channel B receiver and transmitter and the corresponding inputs and outputs.

SRA — Channel A Status Register

SRA[7] — Channel A Received Break — This bit indicates that an all zero character of the programmed length has been received without a stop bit. Only a single FIFO position is occupied when a break is received: further entries to the FIFO are inhibited until the RxDA line returns to the marking state for at least one-half a bit time (two successive edges of the internal or external 1x clock).

When this bit is set, the channel A 'change in break' bit in the ISR (ISR[2]) is set. ISR[2] is also set when the end of the break condition, as defined above, is detected.

The break detect circuitry can detect breaks that originate in the middle of a received character. However, if a break begins in the middle of a character, it must persist until at least the end of the next character time in order for it to be detected.

SRA[6] — Channel A Framing Error — This bit, when set, indicates that a stop bit was not detected when the corresponding data character in the FIFO was received. The stop bit check is made in the middle of the first stop bit position.

SRA[5] — Channel A Parity Error — This bit is set when the 'with parity' or 'force parity' mode is programmed and the corresponding character in the FIFO was received with incorrect parity.

In the special multidrop mode the parity error bit stores the received A/D bit.

SRA[4] — Channel A Overrun Error — This bit, when set, indicates that one or more characters in the received data stream have been lost. It is set upon receipt of a new character when the FIFO is full and a character is already in the receive shift register waiting for an empty FIFO position. When this occurs, the character in the receive shift register (and its break detect, parity error and framing error status, if any) is lost.

This bit is cleared by a 'reset error status' command.

SRA[3] — Channel A Transmitter Empty (TxEMTA) — This bit will be set when the channel A transmitter underruns, i.e., both the transmit holding register (THR) and the transmit shift register are empty. It is set after transmission of the last stop bit of a character if no character is in the THR awaiting transmission. It is reset when the THR is loaded by the CPU or when the transmitter is disabled.

SRA[2] — Channel A Transmitter Ready (TxRDYA) — This bit, when set, indicates that the THR is empty and ready to be loaded with a character. This bit is cleared when the THR is loaded by the CPU and is set when the character is transferred to the transmit shift register. TxRDY is reset when the transmitter is disabled and is set when the transmitter is first enabled, viz., characters loaded into the THR while the transmitter is disabled will not be transmitted.

Preliminary

SRA[1] — Channel A FIFO Full (FFULLA) — This bit is set when a character is transferred from the receive shift register to the receive FIFO and the transfer causes the FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, FFULL will not be reset when the CPU reads the RHR.

SRA[0] — Channel A Receiver Ready (RxRDYA) — This bit indicates that a character has been received and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR, if after this read there are no more characters still in the FIFO.

SRB — Channel B Status Register

The bit definitions for this register are identical to the bit definitions for SRA, except that all status applies to the channel B receiver and transmitter and the corresponding inputs and outputs.

OPCR — Output Port Configuration Register

OPCR[7] — OP7 Output Select — This bit programs the OP7 output to provide one of the following:

- The complement of OPR[7]
- The channel B transmitter interrupt output, which is the complement of TxRDYB. When in this mode OP7 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[6] — OP6 Output Select — This bit programs the OP6 output to provide one of the followng:

- The complement of OPR[6]
- The channel A transmitter interrupt output, which is the complement of TxRDYA. When in this mode OP6 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[5] — OP5 Output Select — This bit programs the OP5 output to provide one of the following:

- The complement of OPR[5]
- The channel B receiver interrupt output, which is the complement of ISR[5]. When in this mode OP5 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[4] — OP4 Output Select — This bit programs the OP4 output to provide one of the following:

- The complement of OPR[4]
- The channel A receiver interrupt output, which is the complement of ISR[1]. When in this mode OP4 acts as an open collector output. Note that this output is not masked by the contents of the IMR.

OPCR[3:2] — OP3 Output Select — This field programs the OP3 output to provide one of the following:

- The complement of OPR[3]
- The counter/timer output, in which case OP3 acts as an open collector output. In the timer mode, this output is a square wave at the programmed frequency. In the counter mode, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state when the counter is stopped by a stop counter command. Note that this output is not masked by the contents of the IMR.
- The 1X clock for the channel B transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the channel B receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

OPCR[1:0] — OP2 Output Select — This field programs the OP2 output to provide one of the following:

- The complement of OPR[2]
- The 16X clock for the channel A transmitter. This is the clock selected by CSRA[3:0], and will be a 1X clock if CSRA[3:0] = 1111.
- The 1X clock for the channel A transmitter, which is the clock that shifts the transmitted data. If data is not being transmitted, a free running 1X clock is output.
- The 1X clock for the channel A receiver, which is the clock that samples the received data. If data is not being received, a free running 1X clock is output.

ACR — Auxiliary Control Register

ACR[7] — Baud Rate Generator Set Select — This bit selects one of two sets of baud rates to be generated by the BRG:

Set 1: 50, 110, 134.5, 200, 300, 600, 1.05K, 1.2K, 2.4K, 4.8K, 7.2K, 9.6K, and 38.4K baud.

Set 2: 75, 110, 134.5, 150, 300, 600, 1.2K, 1.8K, 2.0K, 2.4K, 4.8K, 9.6K, and 19.2K baud.

The selected set of rates is available for use by the channel A and B receivers and transmitters as described in CSRA and CSRB. Baud rate generator characteristics are given in table 3.

Table 3 BAUD RATE GENERATOR CHARACTERISTICS CRYSTAL OR CLOCK = 3.6864MHz

NOMINAL RATE (BAUD)	ACTUAL 16X CLOCK (KHz)	ERROR (PERCENT)
50	0.8	0
75	1.2	0
110	1.759	-0.069
134.5	2.153	0.059
150	2.4	0
200	3.2	0
300	4.8	0
600	9.6	0
1050	16.756	-0.260
1200	19.2	0
1800	28.8	0
2000	32.056	0.175
2400	38.4	0
4800	76.8	0
7200	115.2	0
9600	153.6	0
19.2K	307.2	0
38.4K	614.4	0

NOTE:
Duty cycle of 16X clock is 50% ± 1%.

Preliminary

ACR[6:4]—Counter/Timer Mode and Clock Source Select — This field selects the operating mode of the counter/timer and its clock source as shown in table 4.

ACR[3:0] — IP3, IP2, IP1, IPO Change of State Interrupt Enable — This field selects which bits of the Input Port Change register (IPCR) cause the input change bit in the interrupt status register (ISR[7]) to be set. If a bit is in the 'on' state, the setting of the corresponding bit in the IPCR will also result in the setting of ISR[7], which results in the generation of an interrupt output if IMR[7] = 1. If a bit is in the 'off' state, the setting of that bit in the IPCR has no effect on ISR[7].

IPCR — Input Port Change Register

IPCR[7:4] — IP3, IP2, IP1, IP0 Change of State — These bits are set when a change of state, as defined in the Input Port section of this data sheet, occurs at the respective input pins. They are cleared when the IPCR is read by the CPU. A read of the IPCR also clears ISR[7], the input change bit in the interrupt status register.

The setting of these bits can be programmed to generate an interrupt to the CPU.

IPCR[3:0] — IP3, IP2, IP1, IP0 Current State — These bits provide the current state of the respective inputs. The information is unlatched and reflects the state of the input pins at the time the IPCR is read.

ISR — Interrupt Status Register

This register provides the status of all potential interrupt sources. The contents of this register are masked by the interrupt mask register (IMR). If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the reading of the ISR — the true status will be provided regardless of the contents of the IMR. The contents of this register are initialized to 00_{16} when the DUART is reset.

ISR[7] — Input Port Change Status — This bit is a '1' when a change of state has occurred at the IP0, IP1, IP2, or IP3 inputs and that event has been selected to cause an interrupt by the programming of ACR[3:0]. The bit is cleared when the CPU reads the IPCR.

Table 4 ACR [6:4] FIELD DEFINITION

ACR[6:4]	MODE	CLOCK SOURCE
0 0 0	Counter	External (IP2)
0 0 1	Counter	TXCA — 1X clock of channel A transmitter
0 1 0	Counter	TXCB — 1X clock of channel B transmitter
0 1 1	Counter	Crystal or external clock (X1/CLK) divided by 16
1 0 0	Timer	External (IP2)
1 0 1	Timer	External (IP2) divided by 16
1 1 0	Timer	Crystal or external clock (X1/CLK)
1 1 1	Timer	Crystal or external clock (X1/CLK) divided by 16

ISR[6] — Channel B Change in Break — This bit, when set, indicates that the channel B receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel B 'reset break change interrupt' command.

ISR[5] — Channel B Receiver Ready or FIFO Full — The function of this bit is programmed by MR1B[6]. If programmed as receiver ready, it indicates that a character has been received in channel B and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel B FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[4] — Channel B Transmitter Ready — This bit is a duplicate of TxRDYB (SRB[2]).

ISR[3] — Counter Ready — In the counter mode, this bit is set when the counter reaches terminal count and is reset when the counter is stopped by a stop counter command.

In the timer mode, this bit is set once each cycle of the generated square wave (every other time that the counter/timer reaches zero count). The bit is reset by a stop counter command. The command, however, does not stop the counter/timer.

ISR[2] — Channel A Change in Break — This bit, when set, indicates that the channel A receiver has detected the beginning or the end of a received break. It is reset when the CPU issues a channel A 'reset break change interrupt' command.

ISR[1] — Channel A Receiver Ready or FIFO Full — The function of this bit is programmed by MR1A[6]. If programmed as receiver ready, it indicates that a character has been received in channel A and is waiting in the FIFO to be read by the CPU. It is set when the character is transferred from the receive shift register to the FIFO and reset when the CPU reads the RHR. If after this read there are more characters still in the FIFO the bit will be set again after the FIFO is 'popped'. If programmed as FIFO full, it is set when a character is transferred from the receive holding register to the receive FIFO and the transfer causes the channel A FIFO to become full, i.e., all three FIFO positions are occupied. It is reset when the CPU reads the RHR. If a character is waiting in the receive shift register because the FIFO is full, the bit will be set again when the waiting character is loaded into the FIFO.

ISR[0] — Channel A Transmitter Ready — This bit is a duplicate of TxRDYA (SRA[2]).

IMR — Interrupt Mask Register

The programming of this register selects which bits in the ISR cause an interrupt output. If a bit in the ISR is a '1' and the corresponding bit in the IMR is also a '1', the INTRN output will be asserted. If the corresponding bit in the IMR is a zero, the state of the bit in the ISR has no effect on the INTRN output. Note that the IMR does not mask the programmable interrupt outputs OP3-OP7 or the reading of the ISR.

Preliminary

CTUR and CTLR — Counter/Timer Registers

The CTUR and CTLR hold the eight MSBs and eight LSBs respectively of the value to be used by the counter/timer in either the counter or timer modes of operation. The minimum value which may be loaded into the CTUR/CTLR registers is 0002_{16}. Note that these registers are write-only and cannot be read by the CPU.

In the timer (programmable divider) mode, the C/T generates a square wave with a period of twice the value (in clock periods) of the CTUR and CTLR. If the value in CTUR or CTLR is changed, the current half-period will not be affected, but subsequent half periods will be. In this mode the C/T runs continuously. Receipt of a start counter command (read with A3-A0 = 1110) causes the counter to terminate the current timing cycle and to begin a new cycle using the values in CTUR and CTLR.

The counter ready status bit (ISR[3]) is set once each cycle of the square wave. The bit is reset by a stop counter command (read with A3-A0 = 1111). The command, however, does not stop the C/T. The generated square wave is output on OP3 if it is programmed to be the C/T output.

In the counter mode, the C/T counts down the number of pulses loaded into CTUR and CTLR by the CPU. Counting begins upon receipt of a start counter command. Upon reaching terminal count (0000_{16}), the counter ready interrupt bit (ISR[3]) is set. The counter continues counting past the terminal count until stopped by the CPU. If OP3 is programmed to be the output of the C/T, the output remains high until terminal count is reached, at which time it goes low. The output returns to the high state and ISR[3] is cleared when the counter is stopped by a stop counter command. The CPU may change the values of CTUR and CTLR at any time, but the new count becomes effective only on the next start counter command. If new values have not been loaded, the previous count values are preserved and used for the next count cycle.

In the counter mode, the current value of the upper and lower 8 bits of the counter (CTU, CTL) may be read by the CPU. It is recommended that the counter be stopped when reading to prevent potential problems which may occur if a carry from the lower 8-bits to the upper 8-bits occurs between the times that both halves of the counter are read. However, note that a subsequent start counter command will cause the counter to begin a new count cycle using the values in CTUR and CTLR.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage (except X1/CLK)		2.0			V
V_{IH}	Input high voltage (X1/CLK)		4.0			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except o.c. outputs)	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{OC}	Open collector output leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{CC}	Power supply current				150	mA

NOTES:
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.

Preliminary

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6,7]

PARAMETER	TENTATIVE LIMITS			UNIT
	Min	Typ	Max	
Reset Timing (figure 1)				
t_{RES} RESET pulse width	1.0			μs
Bus Timing (figure 2)[8]				
t_{AS} A0-A3 setup time to RDN, WRN low	10			ns
t_{AH} A0-A3 hold time from RDN, WRN high	0			ns
t_{CS} CEN setup time to RDN, WRN low	0			ns
t_{CH} CEN hold time from RDN, WRN high	0			ns
t_{RW} WRN, RDN pulse width	225			ns
t_{DD} Data valid after RDN low			175	ns
t_{DF} Data bus floating after RDN high			100	ns
t_{DS} Data setup time before WRN high	100			ns
t_{DH} Data hold time after WRN high	20			ns
t_{RWD} High time between READs and/or WRITEs[9,10]	200			ns
Port Timing (figure 3)[8]				
t_{PS} Port input setup time before RDN low	0			ns
t_{PH} Port input hold time after RDN high	0			ns
t_{PD} Port output valid after WRN high			400	ns
Interrupt Timing (figure 4)				
t_{IR} INTRN (or OP3-OP7 when used as interrupts) high from:				
Read RHR (RXRDY/FFULL interrupt)			300	ns
Write THR (TXRDY interrupt)			300	ns
Reset command (delta break interrupt)			300	ns
Stop C/T command (counter interrupt)			300	ns
Read IPCR (input port change interrupt)			300	ns
Write IMR (clear of interrupt mask bit)			300	ns
Clock Timing (figure 5)				
t_{CLK} X1/CLK high or low time	100			ns
f_{CLK} X1/CLK frequency	2.0	3.6864	4.0	MHz
t_{CTC} CTCLK (IP2) high or low time	100			ns
f_{CTC} CTCLK (IP2) frequency	0		4.0	MHz
t_{RX} RxC high or low time	220			ns
f_{RX} RxC frequency (16X)	0		2.0	MHz
(1X)	0		1.0	MHz
t_{TX} TxC high or low time	220			ns
f_{TX} TxC frequency (16X)	0		2.0	MHz
(1X)	0		1.0	MHz
Transmitter Timing (figure 6)				
t_{TXD} TxD output delay from TxC low			350	ns
t_{TCS} TxC output skew from TxD output data	0		150	ns
Receiver Timing (figure 7)				
t_{RXS} RxD data setup time to RXC high	240			ns
t_{RXH} RxD data hold time from RXC high	200			ns

NOTES:
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND). For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. Test condition for outputs: $C_L = 150pF$, except interrupt outputs. Test condition for interrupt outputs: $C_L = 50pF$, $R_L = 2.7K$ ohm to V_{CC}.
8. Timing is illustrated and referenced to the WRN and RDN inputs. The device may also be operated with CEN as the 'strobing' input. In this case, all timing specifications apply referenced to the falling and rising edges of CEN.
9. If CEN is used as the 'strobing' input, this parameter defines the minimum high time between one CEN and the next.
10. Consecutive write operations to the same command register require at least three edges of the X1 clock between writes.

Preliminary

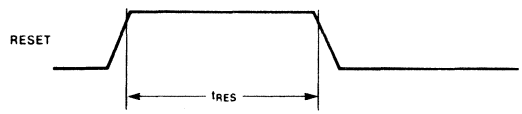


Figure 1. Reset Timing

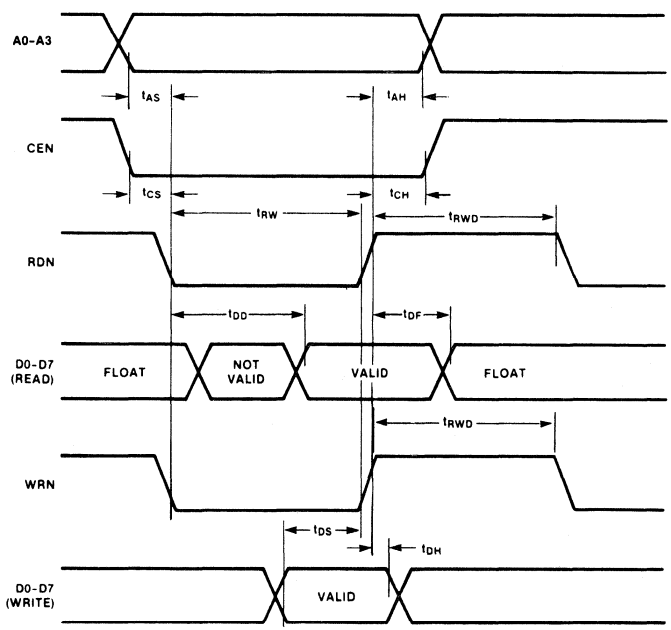


Figure 2. Bus Timing

Preliminary

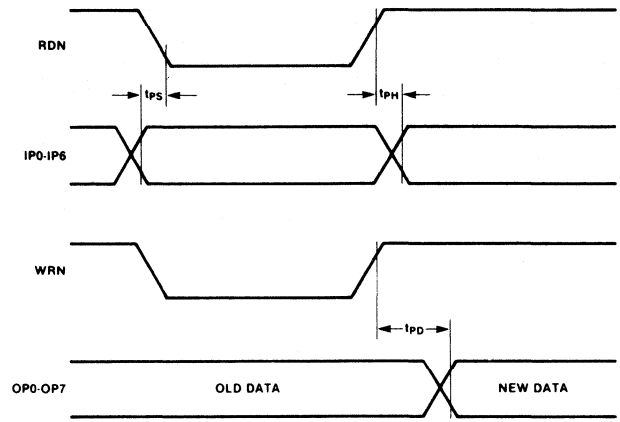


Figure 3. Port Timing

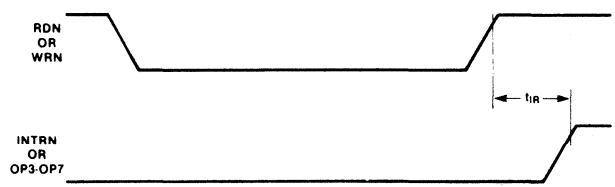
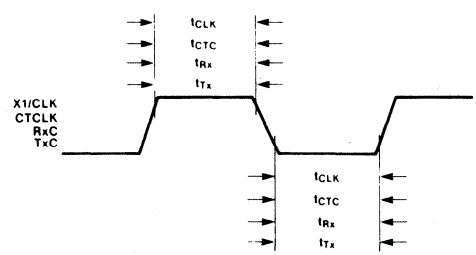


Figure 4. Interrupt Timing



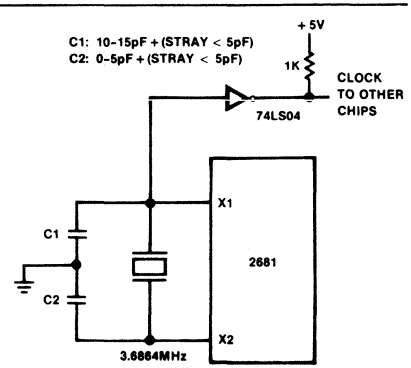


Figure 5. Clock Timing

CRYSTAL SERIES RESISTANCE SHOULD BE LESS THAN 180 OHMS.

Preliminary

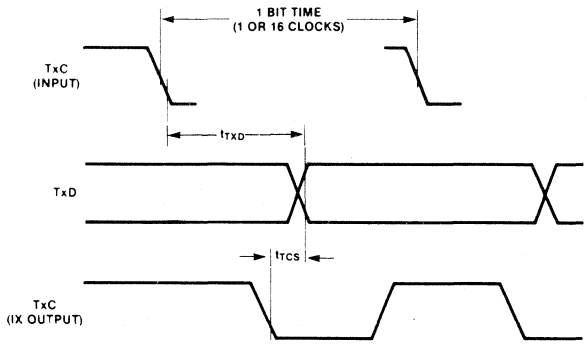


Figure 6. Transmit

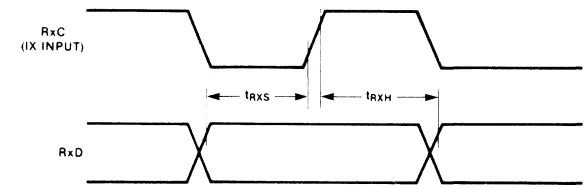
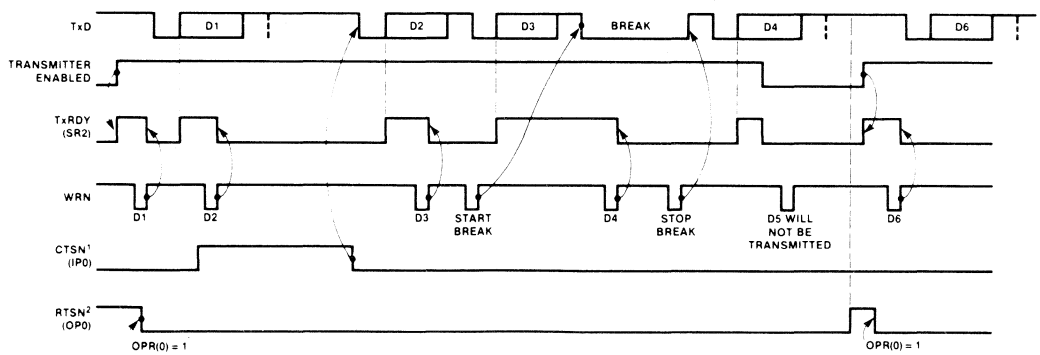


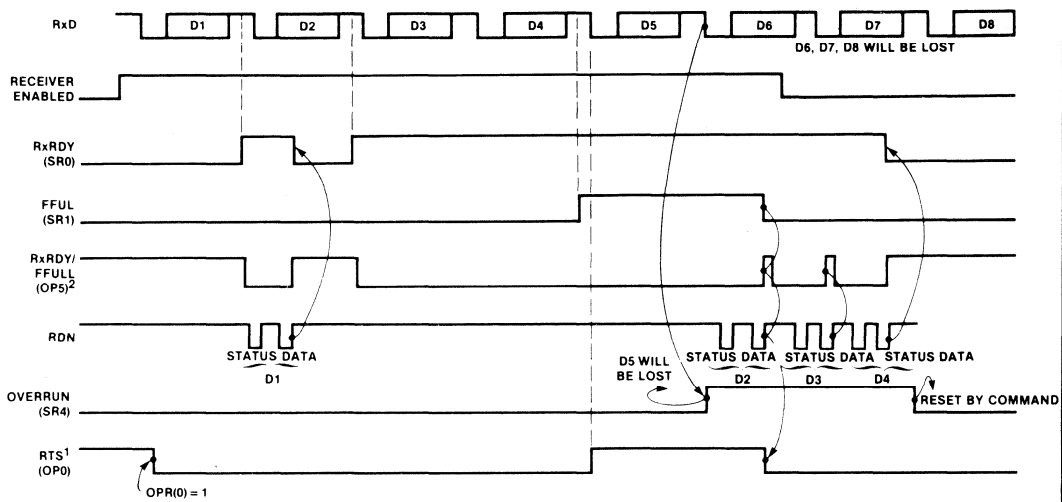
Figure 7. Receive



NOTES
1. TIMING SHOWN FOR MR2(4) = 1.
2. TIMING SHOWN FOR MR2(5) = 1.

Figure 8. Transmitter Timing

Preliminary



NOTES
1. TIMING SHOWN FOR MR1(7) = 1.
2. SHOWN FOR OPCR(4) = 1 AND MR(6) = 0.

Figure 9. Receiver Timing

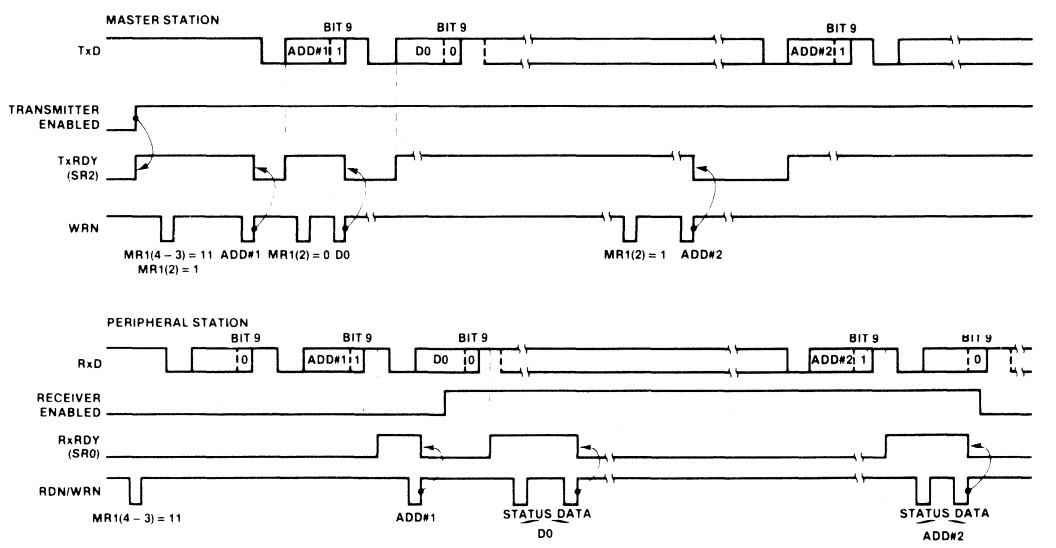


Figure 10. Wake Up Mode

USING THE 2653 POLYNOMIAL GENERATOR AND CHECKER

Originally published by Signetics May 1983

INTRODUCTION

When transferring data via a data communications link using any protocol, the only way to ensure a correct transfer is to perform error checking on the messages being exchanged. Error checking can be accomplished through vertical, longitudinal and cyclic redundancy checks, special character recognition and transparent operating modes. If the error checking is performed correctly, the result is an accurate transfer of data from station to station. The checking technique can be performed by software only, but this may result in a reduction of the maximum channel speed, may reduce the number of channels which can be handled by the CPU, or may limit the supplementary tasks which can be performed by the CPU. The most efficient way to accomplish error checking is to use a combination of hardware and software.

The Signetics 2653 Polynomial Generator and Checker (PGC) is designed to provide the above error checking capability while operating with asynchronous, synchronous or parallel receivers or transmitters at a speed of up to 500K characters per second. The PGC is a device that monitors parallel data transferred between a CPU or memory and a serial receiver/transmitter (R/T, UART, USRT, etc.) or other bus oriented device. Operation is two-way alternate (half-duplex) in that the PGC is selected to receive characters either from the R/T or from the CPU. Full duplex operation is achieved by using two PGCs. A unique feature of the 2653 is its 'character class array', a 128x2 RAM which is used to classify received characters into one of four types - normal, sync/not included, block terminating character, and secondary search character. The received characters may be block checked and/or compared to the special characters preloaded into the character class array. In addition to the block check character (BCC) generation, the PGC is capable of single character detection, two character sequence detection and parity generation and checking. All operating modes are software programmable and can be changed for each application. Figure 1 illustrates the block diagram of the PGC, while figure 2 describes the formats of the registers used to program its operation.[1]

The block check character (BCC), which the 2653 computes from monitoring the 8-bit data bus, takes the form of a cyclic redundancy check (CRC) on specified characters. The CRC is a reliable method of detecting errors in received serial data streams and is employed in almost all synchronous data communications protocols. The PGC can compute the BCC in four modes: BISYNC normal, BISYNC transparent, automatic accumulate, and single accumulate. In each of these modes, one of three error polynomials (CRC-16, CRC-12, and LRC-8) can be selected. In either of the BISYNC modes, the 'intelligence' provided by the character comparison capability within the chip enables it to know which characters to include and which to exclude from the BCC accumulation. Additionally, block terminating characters can be detected as well as the initiation and termination of BISYNC transparent mode. As a result, it can handle character oriented processing for IBM BISYNC, ANSI 3.28, ISO 1745, DEC DDCMP, and other disciplines.

[1]See the 2653 data sheet for full operational description.

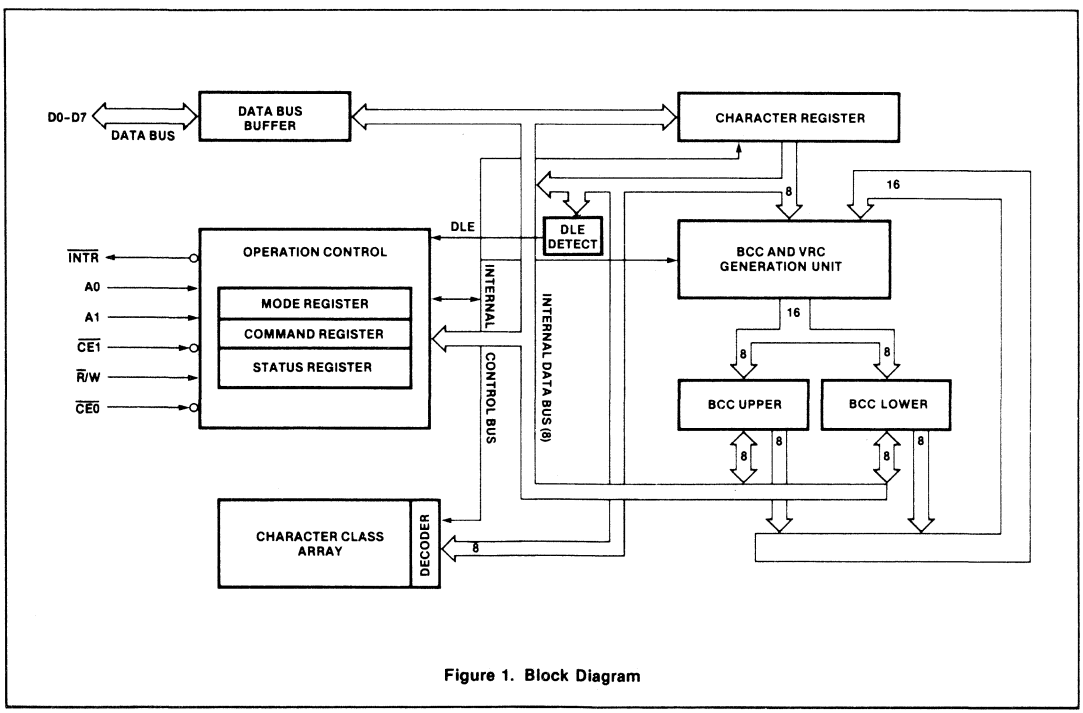
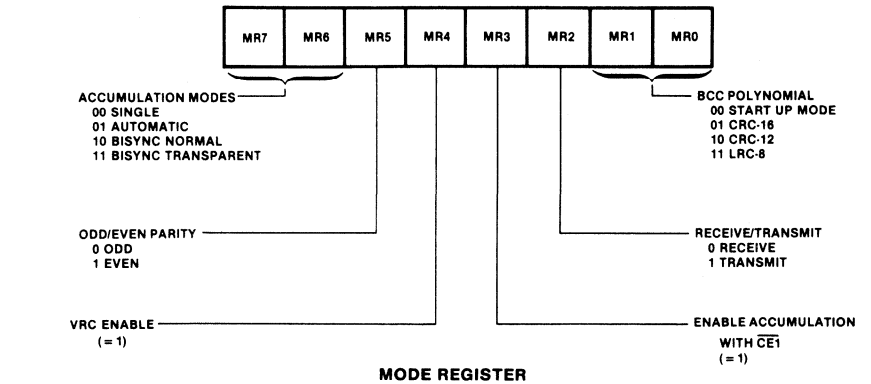
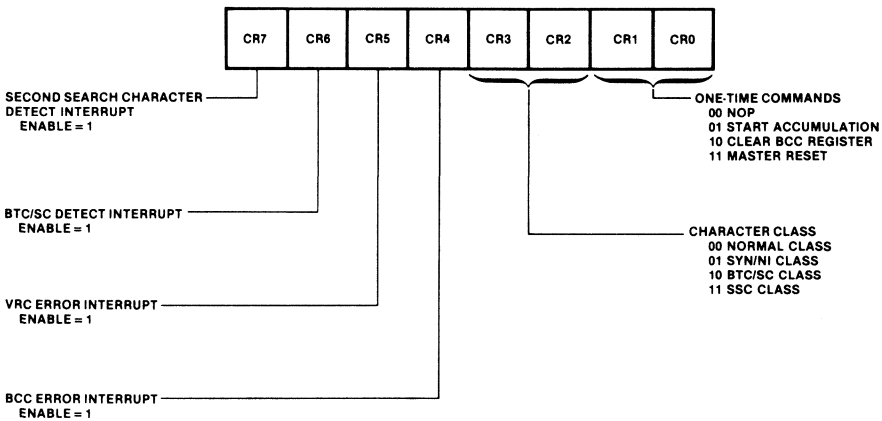


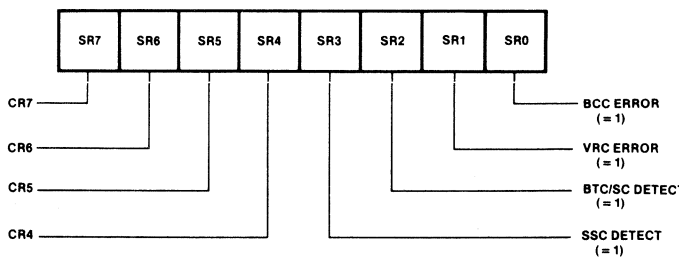
Figure 1. Block Diagram



MODE REGISTER



COMMAND REGISTER



STATUS REGISTER

Figure 2. PGC Register Bit Formats

A companion chip, the Signetics 2661 Enhanced Programmable Communications Interface (EPCI), directly combines with the 2653 to effect a synchronous/asynchronous character oriented communications link. If a complete multi-protocol interface is desired, it can be obtained using the PGC in conjunction with the Signetics 2652 Multi-Protocol Communications Controller (MPCC).

PROTOCOLS

Protocols provide the necessary ground rules to assure the orderly and accurate transfer of data between digital equipments. Data communications protocols are becoming increasingly important as the terminal population increases, distributed processing becomes widespread, and new communications technologies, such as packet switching and satellite links, become commonplace.

The protocols associated with the data communications have been classified into several major levels, or layers, that define various functions and operations. Each level is designed to be functionally independent of the others, but each depends on the correct operation of the previous level to operate. The protocols embodied in these levels range from those that define the physical and electrical links, e.g. RS232C and CCITT V.35, to those which are responsible for functions such as message buffering, code conversion, recognizing and reporting faulty conditions in terminals or lines, communication with the host mainframe, and management of the communication network. These protocols are implemented by software packages such as IBM's Systems Network Architecture (SNA), CCITT's X.25, and DEC's DECnet.

In the remainder of this application note, we shall concern ourselves with data link control protocols (DLC's), which are the sets of rules necessary for effective communications between terminals and computers over conventional communications channels. DLC's are concerned with handling the communications link itself and moving information across it efficiently and accurately.

The basic functions of a DLC are to:

1. Establish and terminate a connection between two stations.
2. Assure message integrity through error detection, requests for retransmission, and positive or negative acknowledgments.
3. Identify sender and receiver through polling or selection.
4. Handle special control functions such as requests for status, station reset, reset acknowledge, start, start acknowledge, and disconnect.

Data link controls can be classified into character oriented protocols (COPs) and bit oriented protocols (BOPs). COPs can be further subdivided into byte control protocols (BCPs) and character count protocols (CCPs).

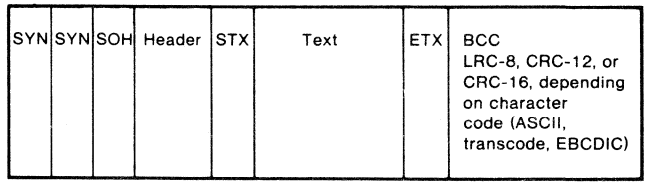


Figure 3. Bisync Text Message Format

BYTE CONTROL PROTOCOLS

In BCPs, a defined set of communication control characters effects the orderly operation of the data link. IBM BISYNC, ANSI 3.28 and ISO 1745 are all byte controlled. Control characters and two character sequences configure and manage the data link between sender and receiver. Control messages or acknowledgements consist of one or two characters while data messages usually contain less than 1,000 characters. For text messages, shown in figure 3, an optional header may precede each text (information) block. The entire message block is error checked based on the information code set used (ASCII, EBCDIC, SBT) and the operational status of a transparent text mode. The transparent text mode is a means of identifying pure data characters from the characters of the information code set. For example, packed BCD, floating point numbers or memory image data would be sent in the transparent mode such that the receiver would not interpret that data as code set characters. Transparent mode is initiated by the sequence DLE-STX and terminated by a DLE followed by a block terminating character (ETX, ETB, ITB, or ENQ).

Byte controlled protocols utilize a stop and wait automatic repeat request (ARQ) which limits operation to two way alternate (half duplex). Each transmitted message block must be acknowledged before the next message may be sent. A negative acknowledgement is achieved by sending a NAK, a positive acknowledgement is sent as an ACK0 or ACK1 for even and odd blocks respectively. The acknowledgement is sent after one or more Block Check Characters (BCCs) have been received and checked (one character for LRC-8, two characters for CRC-12 or CRC-16). Table 1 presents error checking requirements for byte controlled protocols.

For control and acknowledgement messages the receiving processor must detect various single and two character sequences. These are defined in tables 2 and 3.

Table 1 ERROR CHECKING REQUIREMENTS FOR BISYNC/ANSI 3.28

Information Code	No Transparency	Transparency Operating	Transparency Not Operating
EBCDIC	CRC-16	CRC-16	CRC-16
ASCII	VRC-LRC	CRC-16	VRC-CRC-16
SBT	CRC-12	CRC-12	CRC-12

Table 2 COMMUNICATION CONTROL CHARACTERS FOR BISYNC

Mnemonic	Name	Function
SOH	Start of heading	Start of message which is used as heading.
STX	Start of text	Start of any message. Information code characters follow.
ETX	End of text	Signals the end of a text. BCC(s) follow.
ETB	End of transmittal block	Signals the end of a transmittal block. BCC(s) follow.
EOT	End of transmission	If used by the master, it signals the end of a transmission. As a slave response, it indicates an abnormal termination of the transmission (abort). In multipoint systems, it is used by the control station to activate address decoding functions within the tributary stations.
NAK	Negative acknowledgement	Signals back to the master station that the last data block was not accepted. It may also represent a negative response to an initialize sequence, i.e. not ready.
ENQ	Enquiry	Request to send back status, or abort a block of transmitted data. Also used by the master station to end a polling sequence.
ITB	Intermediate block	Blocks of the received message are released to the program via intermediate interrupts for faster processing. BCC(s) follow the ITB.
DLE	Data link escape	Used as leader in control sequences (see table 3).
ACK	Acknowledgement	Used as DLE trailers in control sequences (see table 3).
SYN	Synchronization character	SYN-SYN establishes character synchronization. Inserted automatically into the data stream by the transmitter. Does not enter main storage of the receiver.

Table 3 BISYNC CONTROL CHARACTER SEQUENCES

Mnemonic	Function
DLE-RVI	Indicates to the transmitting station that the receiving station wants to transmit data. Implies acknowledgement of last received block.
DLE-SAK	Indicates to the transmitter that the last message was received free of errors, but the receiver cannot continue.
DLE-STX	Enters transparent text mode. Allows all 256 characters to be used as data.
DLE-EOT	Disconnect sequence on a switched network.
DLE-ETX	End-of-text signal in transparent mode. BCCs follow.
DLE-ETB	End-of-transmittal-block signal in transparent mode. BCCs follow.
DLE-ITB	Intermediate-block-checking signal in transparent text mode. BCCs follow.
DLE-0/1	Used as positive reply to even/odd blocks respectively.
DLE-ENQ	Aborts block of transparent data. BCCs do not follow.
SYN-SYN	Establishes character synchronization. Automatically inserted into the data stream during underrun in normal text mode. Used to maintain synchronization, and to recognize line interruptions. Does not enter main storage of receiver.
DLE-SYN	Automatically inserted into the data stream during underrun in transparent text mode. Used to maintain synchronization, and to recognize line interruptions. Does not enter main storage of receiver.
DLE-WBT	Signals to the transmitting station that the last block was received correctly, but the receiver cannot continue immediately because other operations have to be performed first.
STX-ENQ	Temporary text delay. Abort sequence used by the master station to announce an abnormal termination of the transmission.

Character Count Protocols

Digital Equipment Corporation's DDCMP and its associated versions used by Bell Labs are character count protocols. A character count specifies the number of data characters in the information field of a message: positional significance is used to identify control information in the header of the block which is verified by a separate cyclic redundancy check. There are three control characters in DDCMP (SOH, ENQ, DLE) - each identifies the start of a different type of message. Figure 4 depicts the DDCMP text message format.

A "go back N blocks" type of error control is used in this protocol. Up to 255 blocks may remain outstanding before an acknowledgement is required. This is achieved by separate 8-bit send and receive block counts. When an acknowledgement is sent the received block count indicates the number of message blocks correctly received. This is compared with the send block count. The difference, if any, is the number of blocks that must be retransmitted.

Bit Oriented Protocols

BOPs make use of only two or three specific control characters for operation of the data link. These characters are used to delimit the beginning (FLAG) and end (FLAG, ABORT, GA) of a message frame. Upon receipt of the opening FLAG, positional signficance is used to delineate the bit sequence that follows into prescribed fields, as shown in figure 5. These fields area address, control, information, and frame check sequence. The address, control, and frame check field are fixed length; the information field is variable and may be zero. Examples of BOPs are IBM's Synchronous Data Link Control (SDLC), ANSI's Advanced Data Communication Control Procedures (ADCCP), ISO's High-Level Data Link Control (HDLC), Burroughs' Data Link Control (BDLC), and various other protocols developed by computer mainframe manufacturers. All of the above mentioned protocols are similar and can be treated as subsets of ADCCP. BOPs also utilize a "go back N" type of error control.

2653 FUNCTIONS AND APPLICATIONS

BCC Accumulation

The primary function of the PGC is the accumulation of the BCC for character oriented protocol (BCP and CCP) messages. As described previously, there are four modes of BCC accumulation and each mode can select one of three generating polynomials to compute the BCC(s). The polynomials are $x^{16}+x^{15}+x^2+1$ (CRC-16), $x^{12}+x^3+x^2+x+1$ (CRC-12), and x^8+1 (LRC-8). The four accumulation modes are BISYNC normal, BISYNC transparent, automatic accumulate and single accumulate.

In **BISYNC normal** mode, all characters loaded into the PGC's character register are accumulated except those in the SYN/Not Included class. During receive operations, a detected block terminating character (BTC) will cause the BCC accumulation to stop after the next one (LRC-8) or two (CRC-12 or CRC-16)

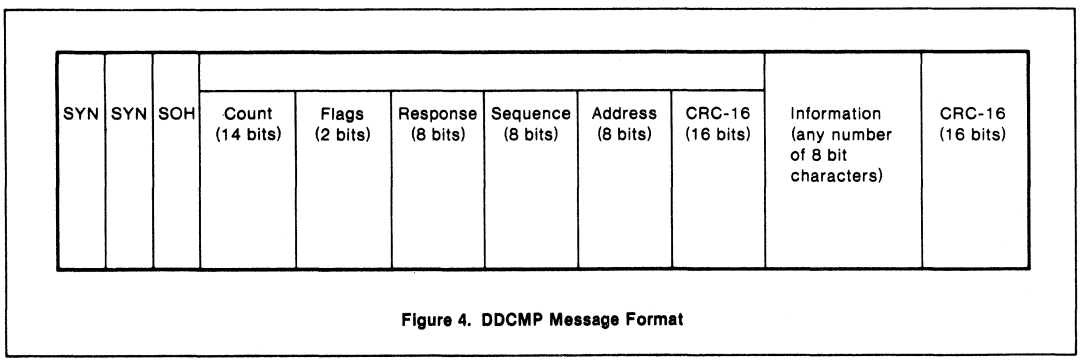


Figure 4. DDCMP Message Format

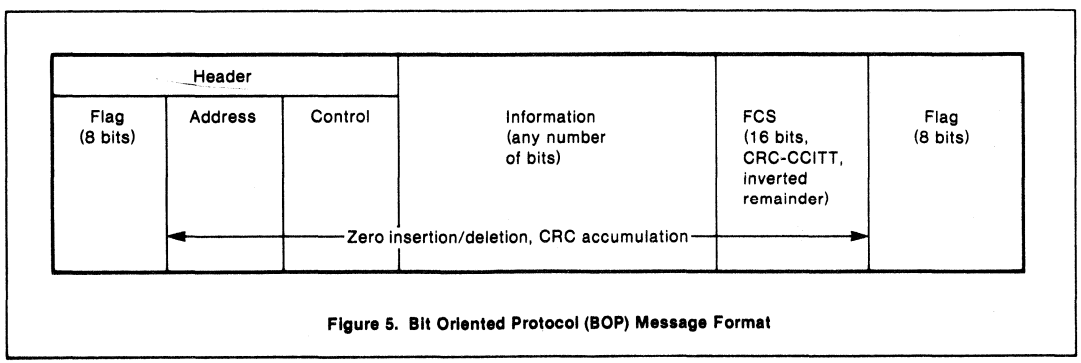


Figure 5. Bit Oriented Protocol (BOP) Message Format

characters have been accumulated. At that time, if the BCC accumulation does not equal zero there has been a block check error. The BCC error bit will be set and an interrupt generated if the corresponding mask bit was enabled. In transmit mode, the BCC accumulation is automatically stopped once the BTC character has been accumulated. The CPU must read the BCC upper and BCC lower (for CRC-12 or CRC-16 only) register(s) and transmit them to the R/T or parallel peripheral. Since the accumulation has been stopped, the transfer of the first BCC to the R/T will not effect BCC lower. This assures that the second BCC will be correct when it is read by the CPU.

Note that BCCs are not checked against the character class array nor are they compared to the DLE ROM. This prevents false character detections when transmitting or receiving BCCs.

Second search character (SSC) detection is enabled in BISYNC normal allowing a two character communication control sequence such as DLE-STX to be detected.

In **BISYNC transparent** mode characters excluded from the BCC accumulation are the first DLE of a DLE-DLE pair, the DLE of a DLE-BTC pair, or DLE-SYN sequences (the SYN is also excluded).

In receive and transmit modes, the termination of BCC accumulation works exactly as in BISYNC normal. except that the BTC must be immediately preceded by an odd number of DLEs to be properly identified.

Second search character detection is not enabled in BISYNC transparent since DLE-SSC sequences are only valid in BISYNC normal mode.

In **Automatic accumulate** mode all characters loaded into the character register are accumulated; BTC and SSC detection is enabled and the BCC accumulation is not automatically terminated. The CPU must use single accumulate mode to stop the accumulation. When in receive mode, the BCC error bit is set/reset after accumulating each character so that the CPU must examine this bit after the last character is accumulated.

Examples of use of the automatic accumulate mode are a system where the R/T (2651/2661) operates with DLE/SYN stripping or in support of character count protocols such as DDCMP.

In **Single accumulate** mode all characters are accumulated, but only after an accumulate command is given by the CPU. If not given, the BCC accumulation is stopped. Operation in this mode is otherwise identical to automatic accumulate. Single accumulate mode can be used to selectively accumulate characters under CPU control or to accumulate characters that were unintentionally excluded in one of the other modes.

Figures 6 and 7 illustrate the operation of the 2653 on various types of text and control messages.

Some Other Applications

The PGC can be employed in a variety of applications other than a dedicated BCC generator for a single channel. For example, it can be multiplexed among several data channels, used as a programmable character comparator or it can be used to check parity on a system address or data bus. A brief description of each of these applications is given below.

a. MULTIPLEXED PGC

One PGC may be time-shared among a few R/T's if the CPU saves and restores the mode register and partial BCC result in the BCC registers. These registers are accessed via CE1. There must be separate save area for each R/T (serial channel) and a channel pointer indicating the last R/T that transferred or received a data character (see figure 8).

The loading of the BCC registers will clear SR0-SR3 and all previously detected special characters, i.e., DLE, BTC/SC, BCC (BISYNC modes). The BCC accumulation will start again when the next character is loaded into the character register in all accumulation modes except single. That mode requires a start accumulation command.

b. CHARACTER COMPARATOR

The PGC can be used as a programmable data bus character comparator which monitors data bus character transfers (CPU to peripheral, CPU to CPU, CPU to memory, memory to peripheral via DMA). The user selectively loads the character class array with BTC/SC and SSC characters to be compared. Status bits will be set and an interrupt can be generated upon SC and DLE-SSC detection. A match on one to 128 different characters or DLE-SSC sequences can be programmed.

Figure 9 depicts an arrangement where the DMA controller or slave CPU handles data bus transfers, the PGC interrogates the data bus, and the host CPU responds to PGC interrupts.

c. BUS PARITY CHECKER

The PGC can be used to check the parity of transactions on a system's data bus. The processor first writes control information into the PGC via the CE1 pin. All other bus operations are then checked for parity with external address decoding used to generate an active low CE0. Bus parity checking is useful in data transfers between CPU and peripherals or memory and CPU. Some computers check parity on both halves of a 16-bit word during all system bus transfers.

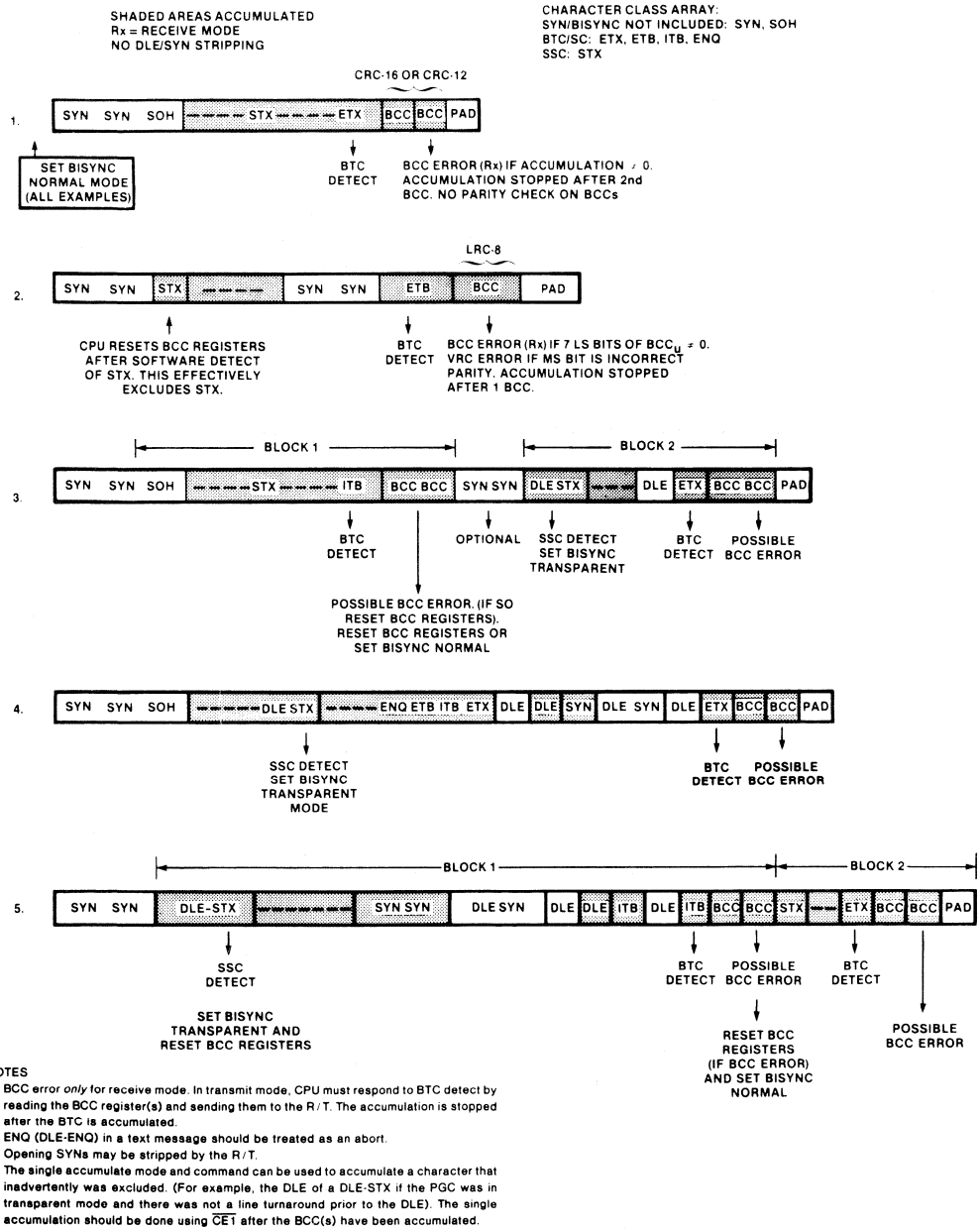
MULTI-PROTOCOL SYNCHRONOUS CHIP SET

The Signetics 2652 Multi-Protocol Communications Controller (2652), originally targeted for bit oriented protocols and DDCMP, can send and receive EBCDIC, ASCII, and SBT data. However, the 2652 doesn't support many of the functions of byte controlled protocols. In particular, the 2652 has no way of knowing which characters to include or exclude in the BCC accumulation. This makes the on board CRC-16 generator/checker useless for BISYNC. Furthermore, there are no provisions in the 2652 for transparent mode DLE handling, special character detection or two character sequence detection. But the PGC encompases all of these missing functions! Thus, the 2652 - 2653 combination can totally support character controlled protocols as well as bit oriented and character count disciplines. The PGC can be used for single character compares in SDLC/HDLC or DDCMP applications to reduce software overhead.

As shown in figure 10, only a single inverter is required to interface the 2652 and 2653 such that 2652 data bus transfers are monitored.

A BISYNC/ASYNC CHIP SET

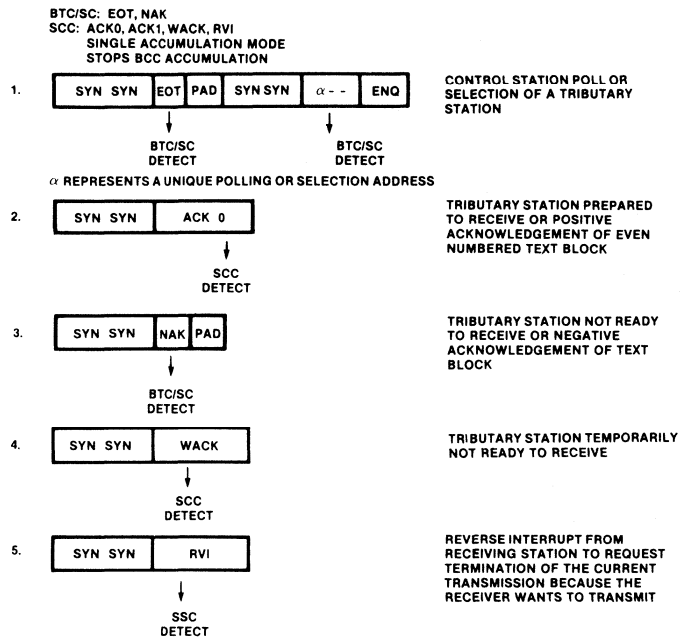
Although the 2653 complements any R/T in the support of character controlled protocols it is optimized for use with the Signetics 2661 Enhanced Programmable Communications Interface (EPCI). That device is a USART with on chip baud rate generator that has special features for BISYNC. There are two loadable SYN



NOTES

1. BCC error *only* for receive mode. In transmit mode, CPU must respond to BTC detect by reading the BCC register(s) and sending them to the R/T. The accumulation is stopped after the BTC is accumulated.
2. ENQ (DLE-ENQ) in a text message should be treated as an abort.
3. Opening SYNs may be stripped by the R/T.
4. The single accumulate mode and command can be used to accumulate a character that inadvertently was excluded. (For example, the DLE of a DLE-STX if the PGC was in transparent mode and there was not a line turnaround prior to the DLE). The single accumulation should be done using $\overline{CE1}$ after the BCC(s) have been accumulated.

Figure 6. Examples — Bisync Text Message BCC Accumulation



NOTES
1. BCC accumulation should be ignored for control messages. This can be effected by single accumulate mode without single accumulate commands.
2. Characters programmed as SSCs should be the binary equivalent of the second character of the DLE-SSC sequence.

Figure 7. Examples — Bisync Control Messages

registers and a loadable DLE Register in the EPCI. Figure 11 is a schematic showing the 2653 and 2661 interfaced to an 8-bit CPU.

A transparent operating mode causes the EPCI to automatically change the detected synchronization sequence and underrun linefill from SYN-SYN to DLE-SYN. This is necessary to prevent an unrecoverable problem at the receiver. If a USART sent or received the normal mode SYN-SYN sequence it would be interpreted as transparent data rather than actual synchronization information.

Another transparent mode function is detecting and stripping received DLE's. Normally a software job, this task is completely and properly handled by the 2661. The DLE Detect status bit is even automatically reset at the proper time.

A Send DLE command in the 2661's transmitter can be used to prevent a possible underrun between the DLE and a subsequent control character. Such an underrun would cause an incorrect control sequence to be transmitted. For example, consider an underrun between a DLE-STX, the sequence used to enter transparent mode. The transmitted sequence becomes DLE-SYN-SYN-STX. But a DLE-SYN is illegal unless transparent mode has been entered. Furthermore, the STX would set normal text mode not transparent mode.

FLOW CHARTS FOR BISYNC OPERATION

Figures 12 through 15 illustrate functional flow charts for the operation of the 2653 - 2661 pair in BISYNC. The intent of these flow charts is to illustrate the procedures required when receiving and transmitting BISYNC text messages in both normal and transparent modes of operation. It is not implied that the actual software program to handle these tasks necessarily follow the flow charts step by step. In an actual application, an interrupt driven structure would be more appropriate. Assumptions are half-duplex operation (normal for the BISYNC protocol) and use of the EBCDIC code.

The receive flow, figure 12, starts with initialization of the PGC and EPCI for the normal mode. Modem handshaking is then performed. Upon detection of carrier, indicated by assertion of the 2661's DCD status bit, the receiver is enabled, the PGC is setup for receive, and miscellaneous flags are reset. Data is then read from the 2661 receive holding register and acted upon according to the BISYNC protocol. The

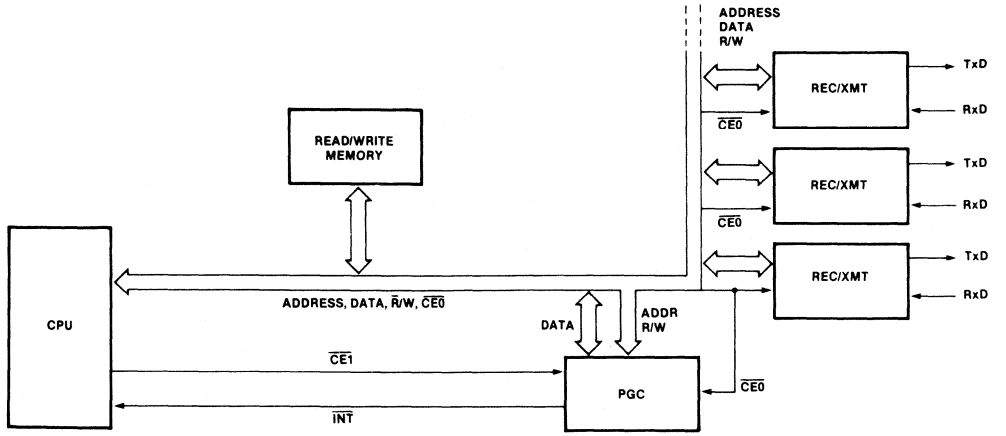


Figure 8. PGC Services Multiple Receivers/Transmitters

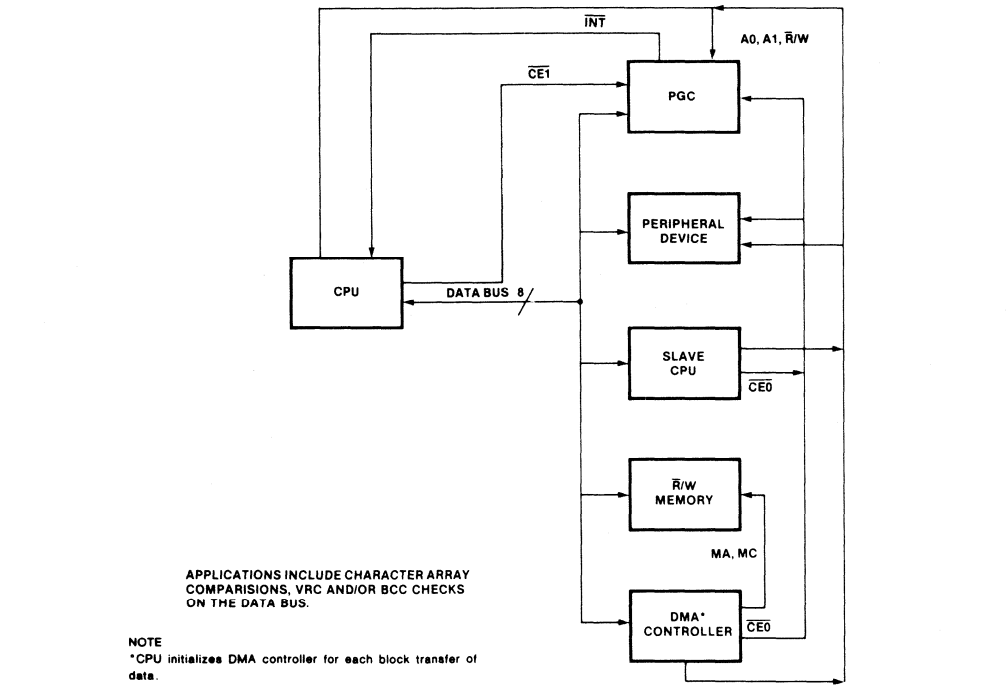


Figure 9. PGC Data Bus Monitoring with DMA Transfers

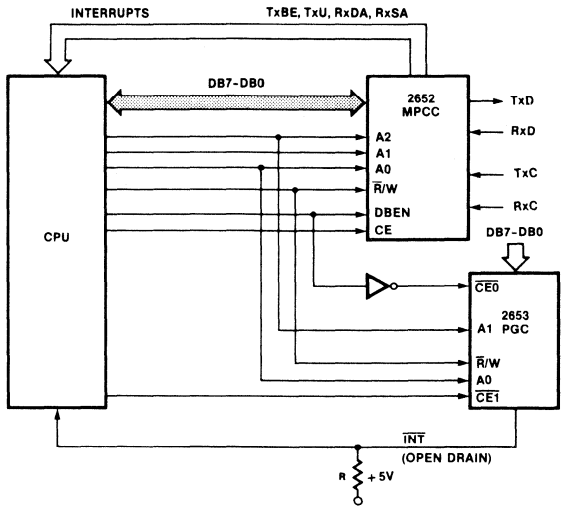


Figure 10. 2652/2653 Interface — Typical Protocols: BISYNC, DDCMP, SDLC, HDLC

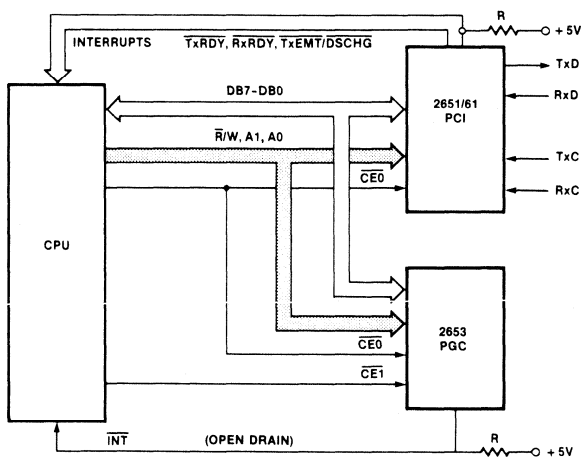


Figure 11. 2651 or 2661/2653 Interface

PGC status flags are utilized to determine if and when the transmission switches to transparent mode, and to determine the receipt of a block terminating character (BTC). The two characters following the BTC are the Block Check Character. After these are received, the PGC status register is examined to determine if a BCC error has occured.

The data stored in the buffer will be stripped of all sync characters. DLEs are not stripped. Although the 2661 includes a DLE stripping capability, this feature is not employed because the DLEs must be 'seen' by the PGC in order for it to accumulate the BCC correctly. The CPU must remove the extraneous DLEs which may be imbedded in a transparent block of text.

The transmit flow chart, figure 13, operates on a block of data placed in a buffer area by the controlling CPU. This data must include the SYNs to be sent at the initiation of transmission and the DLEs that form part of a two character control sequence. DLEs in a transparent block of text need not be doubled up - the EPCI will automatically add a DLE if one is loaded into its THR while operating in transparent mode. A character counter assists the software to determine when a DLE is really part of a BTC (in transparent mode).

After initialization of the PGC and EPCI and establishment of the modem connection, the data is pulled from the buffer and transmitted. If a DLE is detected in the data stream, and that character is part of a two character control sequence, the 'send DLE' feature of the PCI is used to avoid underrun between the two characters. Since the DLE is not transferred to the EPCI via the data bus, this requires that an extra DLE be accumulated in the PGC. This is done by use of the PGC's capability to accumulate characters loaded via CE1. When a BTC is detected by the PGC, the two BCC characters are read from the BCC registers and transferred to the EPCI for transmission.

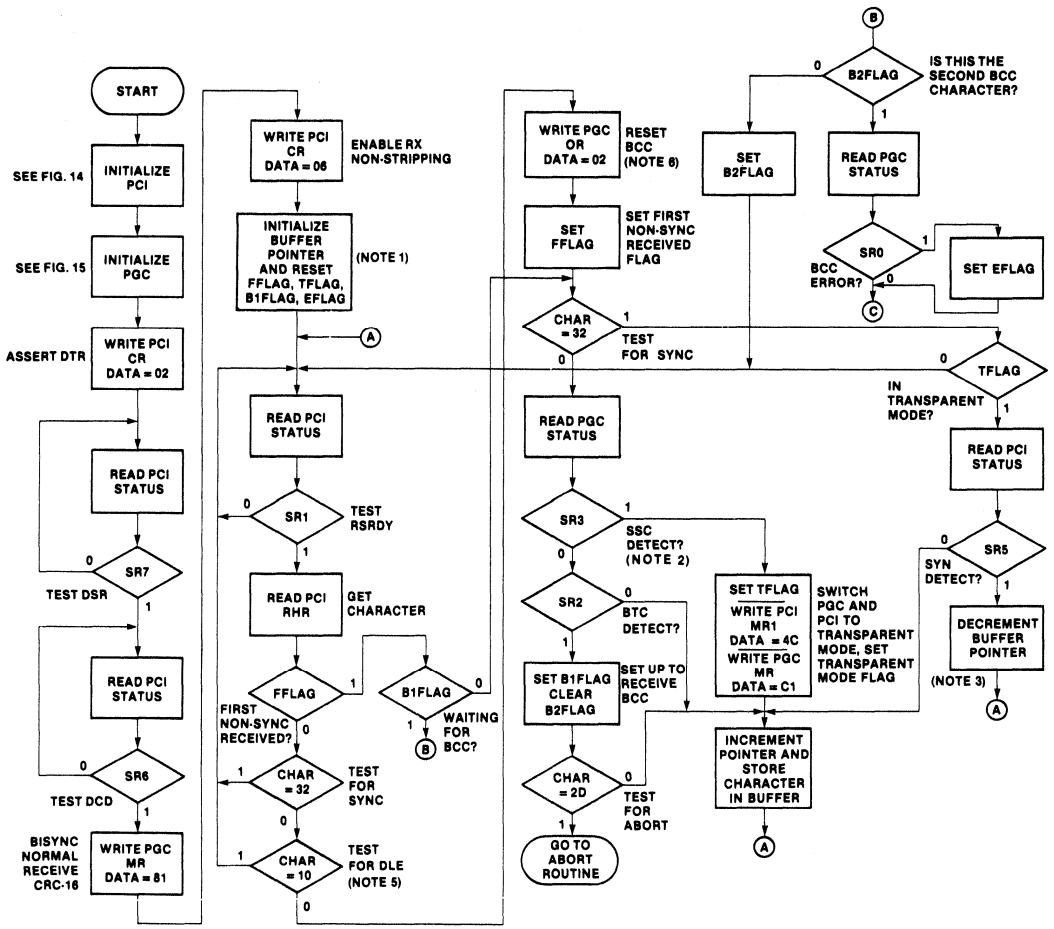
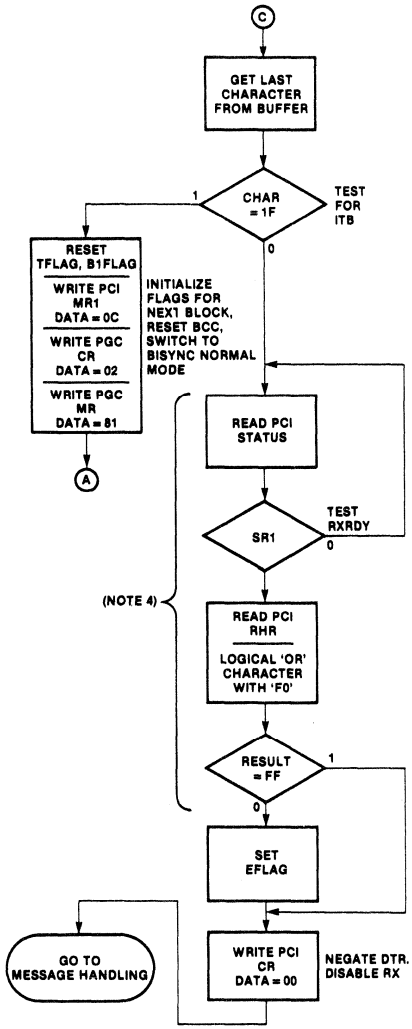


Figure 12. Bisync Receive Flow Chart



BISYNC RECEIVE FLOW CHART NOTES

1. FFLAG = the first non-sync character has been received

   TFLAG = operating in transparent mode

   EFLAG = BCC or PAD error.

   B1FLAG = Received block terminating character (BTC). Awaiting BCC.

   B2FLAG = Received first BCC character. Awaiting second BCC.
2. SSC detect is disabled by PGC while in transparent mode.
3. Pointer is decremented to overwrite previously stored 'DLE' which was part of a 'DLE-SYN' line fill.
4. Test for closing PAD of at least four ones at end of message.
5. If first non-sync character is a 'DLE', the message will start with 'DLE-STX' (transparent mode). FFLAG is not set in this case since both these characters are excluded from the accumulation.
6. First non-sync character of a new message, or first two if message starts in transparent mode, are excluded from the BCC accumulation.

Figure 12. Bisync Receive Flow Chart (Continued)

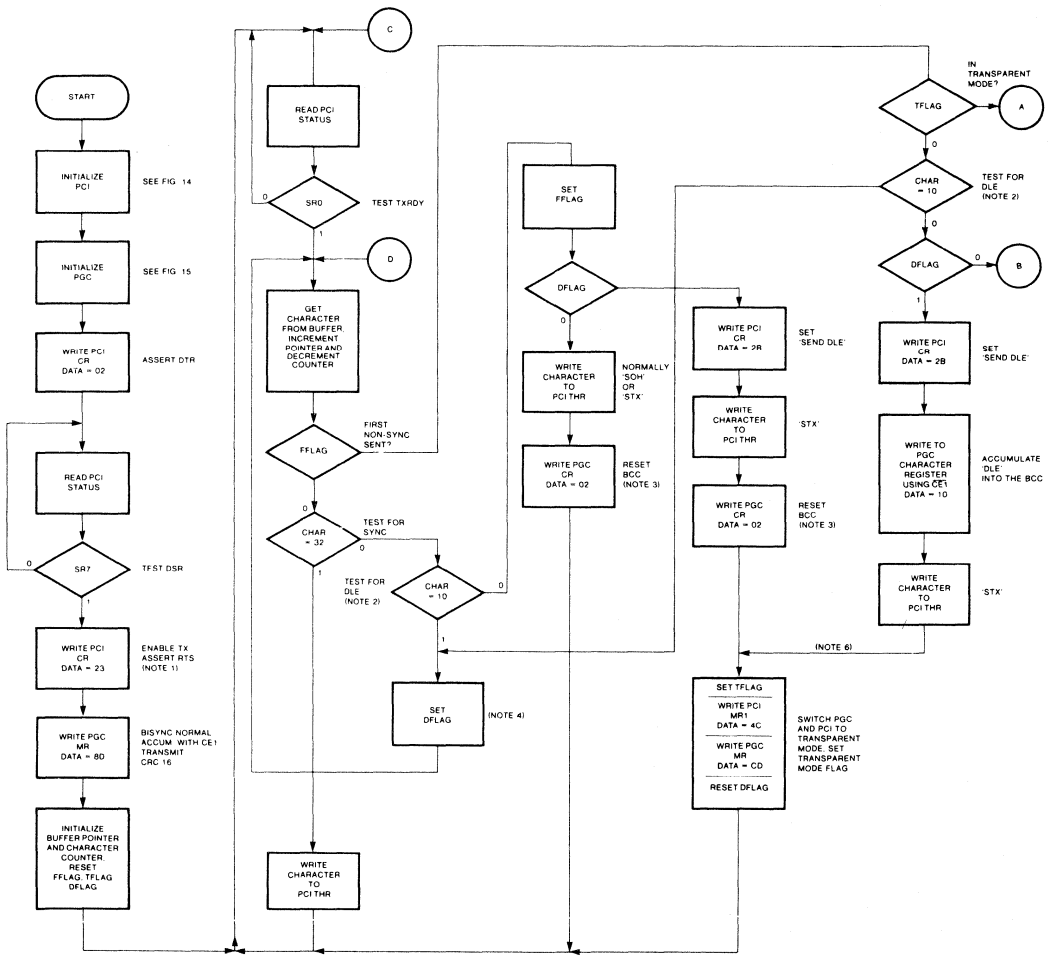
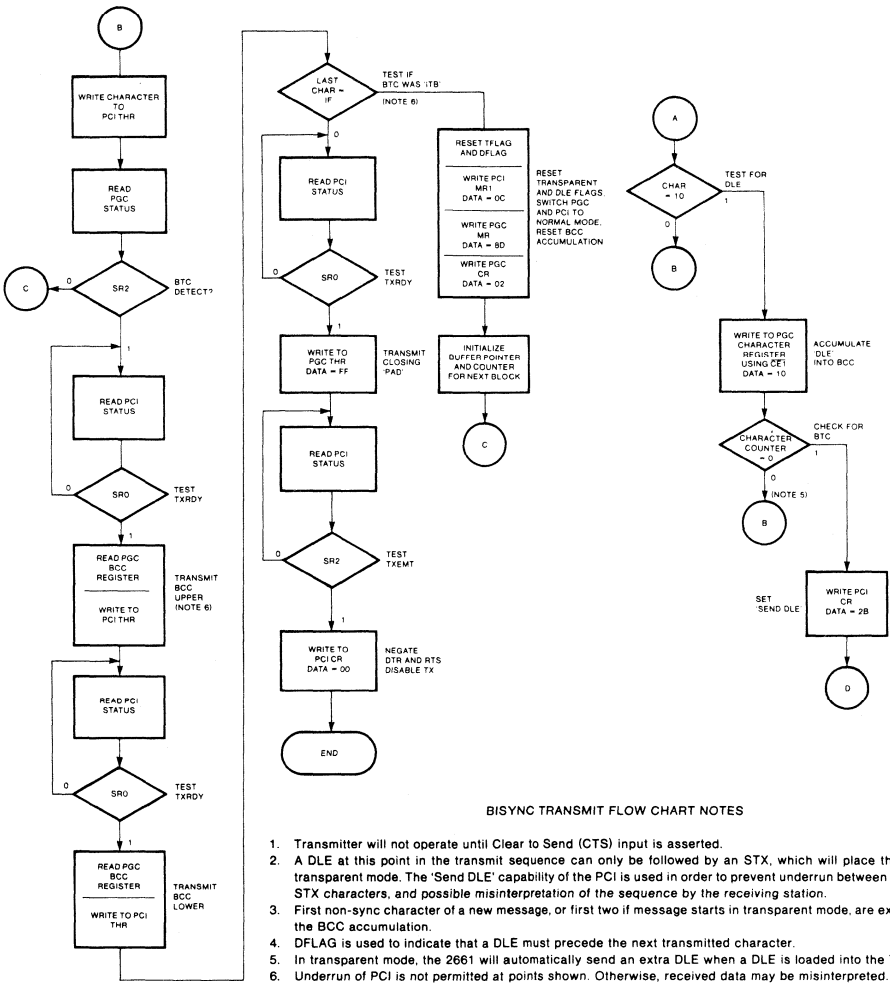


Figure 13. Bisync Transmit Flow Chart



BISYNC TRANSMIT FLOW CHART NOTES

1. Transmitter will not operate until Clear to Send (CTS) input is asserted.
2. A DLE at this point in the transmit sequence can only be followed by an STX, which will place the system in transparent mode. The 'Send DLE' capability of the PCI is used in order to prevent underrun between the DLE and STX characters, and possible misinterpretation of the sequence by the receiving station.
3. First non-sync character of a new message, or first two if message starts in transparent mode, are excluded from the BCC accumulation.
4. DFLAG is used to indicate that a DLE must precede the next transmitted character.
5. In transparent mode, the 2661 will automatically send an extra DLE when a DLE is loaded into the THR.
6. Underrun of PCI is not permitted at points shown. Otherwise, received data may be misinterpreted.

Figure 13. Bisync Transmit Flow Chart (Continued)

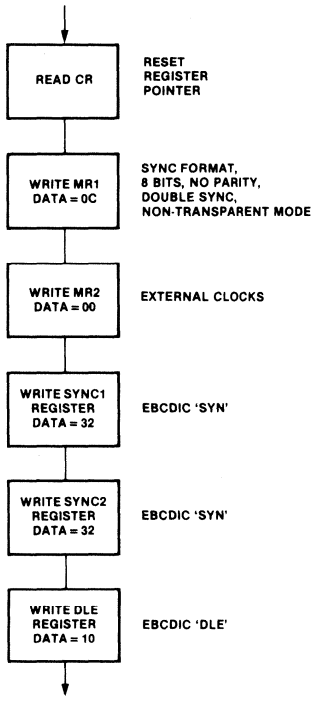


Figure 14. 2661 Initialization

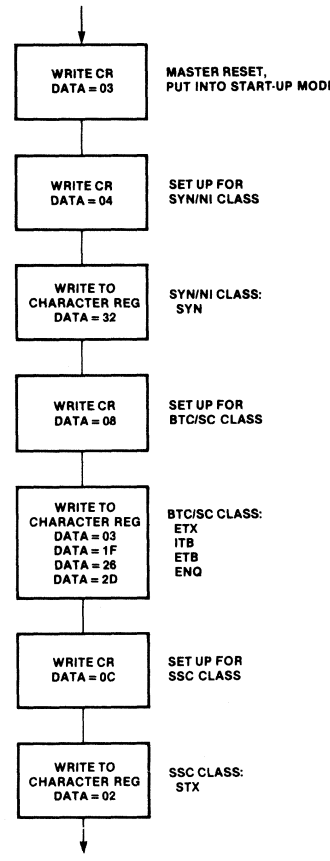


Figure 15. 2653 Initialization

2661 OPERATING MODE SWITCHING PROCEDURES

INTRODUCTION

This application note describes procedures for switching the operating mode of the Signetics' 2661 Enhanced Programmable Communications Interface (EPCI) from echoplex or remote loopback mode to normal operation and vice-versa.

ECHOPLEX (AUTOMATIC ECHO) MODE TO NORMAL OPERATION

The echoplex operation is initiated by setting command register bits CR7:CR6 = 01, and CR2 (receiver enable bit) = 1. Echoplex operation is terminated by resetting CR2 to zero. To ensure the proper transmission of the last received character, no change of operating mode should be made until the end of that character. However, if mode switching is necessary in certain applications, the following procedure is recommended to ensure no garbling on the last transmitted character. Two potential problems may arise: the calculated parity instead of the received parity may be transmitted, and data rate may be shortened or lengthened.

The procedure provides the necessary handshaking to avoid these potential problems by making use of the $\overline{\text{TXEMT}}$/$\overline{\text{DSCHG}}$ pin or of the status register bit 2, SR2, to indicate the end of the parity bit or the first stop bit, depending on whether one or two stop bits are selected (MR17:MR16 = 01 or 11). The procedure causes $\overline{\text{TXEMT}}$/$\overline{\text{DSCHG}}$ to be driven to its active state only at the completion of the last character, as shown in figure 1.

The recommended sequence of operation is as follows:

1. Wait for RXRDY (either RXRDY interrupt or status read). This is necessary for the assembly of the last character to be completed and to ensure the transfer of this character to the transmitter.
2. Enable the transmitter by setting CR0 to one.
3. Disable the receiver by setting CR2 to zero.
4. Wait for TXEMT (either $\overline{\text{TXEMT}}$/$\overline{\text{DSCHG}}$ interrupt or status read). At this point, the parity bit or the first stop bit (if two stop bits are selected) has been sent out.
5. Change mode from echoplex to normal.
6. Load new character into the transmit holding register, THR. Further communication between the 2661 chip and the CPU will resume as normal - that is, $\overline{\text{TXRDY}}$ is driven active to indicate that the THR is available for new data and $\overline{\text{TXEMT}}$ is driven active upon underrun condition.

Note that the $\overline{\text{TXEMT}}$ pin is not driven active in echoplex mode. It is optionally driven active when the above steps are followed, particularly the transmitter being enabled as indicated in step 2. Because the transmitter relies on CR0 = 1 and CR2 = 0 to drive $\overline{\text{TXEMT}}$ active, it is necessary to set CR0 to zero in echoplex mode if it is desired not to drive $\overline{\text{TXEMT}}$ active. CR0, transmitter enable, is ignored for data transmission in echoplex mode. It is, however, used to determine whether $\overline{\text{TXEMT}}$ should be driven active.

If frequent mode switching is anticipated and it is desired to drive $\overline{\text{TXEMT}}$ active, step 2 of the above procedure could be skipped, provided that the echoplex operation is initiated by enabling both the receiver and the transmitter - that is, CR2:CR0 = 11.

The $\overline{\text{TXEMT}}$ timing shown above is only applicable when switching modes. Note that in normal operation, $\overline{\text{TXEMT}}$ is driven active at the beginning of the last data bit or parity bit upon underrun condition.

$\overline{\text{TXEMT}}$/$\overline{\text{DSCHG}}$ OPERATION FOR ECHOPLEX MODE SWITCHING

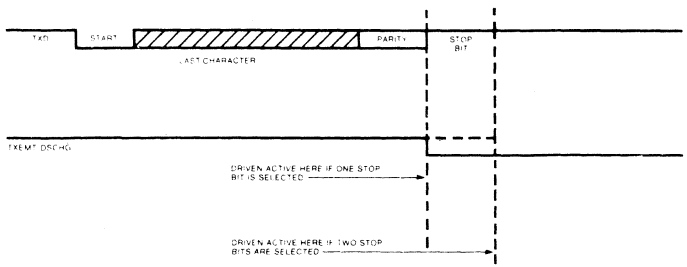


Figure 1

SWITCHING FROM NORMAL TO ECHOPLEX OR REMOTE LOOPBACK

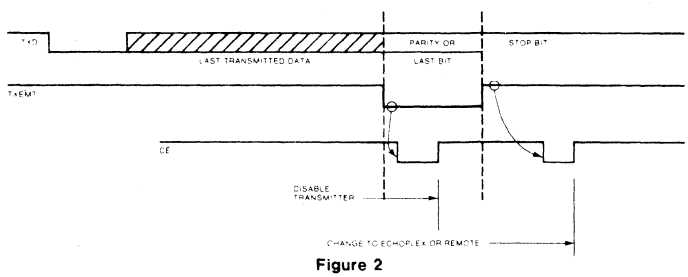


Figure 2

REMOTE LOOP BACK MODE TO NORMAL OPERATION

The procedure is similar to the procedure for echoplex to normal, with the following exceptions:

1. No handshaking with RXRDY is required.
2. During step 3 of the previous procedure, CR2 goes to zero, and CR7:CR6 should be simultaneously changed from 11 to 01 (remote to echoplex). This is necessary because the logic implemented to drive $\overline{\text{TXEMT}}$ active relies on echoplex information. However, this requirement does not need additional service from the controller because remote-to-echoplex switching is done at the same time as disabling the receiver.

NORMAL OPERATION TO ECHOPLEX OR REMOTE

To avoid garbling the last transmitted data, a mode switch from normal operation to echoplex or remote operation should be performed as follows:

1. Wait for TXEMT (either $\overline{\text{TXEMT}}$/$\overline{\text{DSCHG}}$ interrupt or status read) to be asserted.
2. Disable the transmitter by setting CR0 to zero.
3. Wait for TXEMT to be negated.
4. Change the mode from normal operation to echoplex or remote.

The timing is illustrated in figure 2.

DATA COMM PROTOCOLS

Originally published by Signetics April 1983

PROTOCOLS

PURPOSE OF PROTOCOLS

Data communications protocols are standard procedures and conventions that govern the transfer of data among communicating data processing machines. The machines may be computers sending data to printers, keyboards transferring data to controllers or disks sending and receiving data to host processors. The ultimate purpose of all data communications protocols is to avoid data loss and to provide data security against unauthorized recipients. Protocols are also called line protocols, line disciplines, conventions, and line formats.

In its simplest form, a protocol is a list of rules to be followed when transferring data. When the transfer is within a single machine, protocol is unimportant because the data transfer is controlled by the design of the machine. However, transfers between different machines must occur in an environment which cannot be controlled by the machine design. For example, electrical interference can be minimized inside the machine by prudent design practices and by shielding. Unauthorized access to data can be circumvented by the physical design of the machine.

FUNCTIONS

In general, protocols defined for data communications perform two functions:

1) Establish a connection between sender and receiver.
2) Transfer the information reliably.

Establishing a connection might be as simple as addressing a peripheral device, such as a printer, in the input/output space of a microprocessor, or it might be as complex as formatting a series of messages to create a virtual channel between two devices at different nodes of an international data communications network.

Transfer of the information reliably includes the detection and sometimes the correction of errors. Even with an excellent protocol, it is still impossible for all messages to arrive at their destinations without degradation. Thus the detection of errors is specified in terms of a statistical probability of error after the reception of a certain number of bits that have been transmitted, called error rate. The goal is to achieve the lowest error rate possible within reasonable cost limits.

Protocols perform the two functions just described through a series of activities, as follows:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.
5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

If a program or data processing machine using a protocol only needs to specify the nature of the transfer but not participate in controlling the transfer, the protocol is called transparent. If the user must participate in the transfer beyond specifying the source and destination addresses, the location of the data at the source and the intended buffer address at the destination, the protocol is called virtual. Protocols currently in use vary in the amount of transparency incorporated in the definition. The newer the protocol, the more transparent it is likely to be.

SYNCHRONOUS AND ASYNCHRONOUS

Protocols may also be classified by the means in which they allow for the data to be transmitted. If the protocol provides for coordination of the sending and receiving station to effect the transfer of data, the protocol is called synchronous. If the protocol allows the sending station to send data at random times with or without the prior notification of the receiver, the protocol is called asynchronous. In particular, these differences appear in the rules defined for data transfers. Synchronous protocols usually include special characters, called sync characters, to allow for the sending and receiving stations to achieve synchronism in time. The achievement of synchronism in time assumes that the stations have achieved synchronism with word boundaries. Asynchronous protocols do not require synchronism in time, and word boundaries are automatically defined by a fixed time interval that precedes and follows each character.

Since there are many protocols in use, some defined by national and international standards organizations, others defined by digital equipment manufacturers, the various characteristics of protocols are by no means standard except within the domain of a specific manufacturer. Thus some asynchronous protocols have a single bit at the beginning of a character transmission, others have several. Some synchronous protocols have a single synchronization character at the beginning of a message, others have several. Even among the users of the same number of sync characters, different manufacturers may use different characters for synchronization.

RELIABILITY

Among the rules established by a protocol are provisions for detecting and recovering from error conditions. Error conditions may arise from the presence of noise in the transmission channel or from malfunction-

ing equipment. In any event, the protocol must provide a means for detecting the occurrence of an error. One of the chief means of detecting errors is to provide check bits. These are extra bits added to the transmitted data which provide clues to the receiver concerning the nature of the transmitted data. Using these clues, the receiving station can detect the occurrence of an error and take the appropriate recovery action. Since the check bits effectively repeat a part of the data, they are called redundant bits.

Among the methods of adding redundancy to the message are the addition of parity bits and characters. If every character transmitted is followed by a parity bit, the check is called vertical parity checking or vertical redundancy checking (VRC). If a special parity character is added in which each bit of the special character creates a parity check on the corresponding bit for each of the preceding characters in the message block the check is called a longitudinal parity check or longitudinal redundancy check (LRC). Algebraic techniques are also used to generate check characters. When the entire message is viewed as a polynomial, it is possible to consider dividing the message polynomial by another fixed polynomial and using the remainder as the check character. The redundancy check using this technique is called cyclic redundancy checking (CRC) after the class of polynomials used to generate the remainder.

The use of the various reliability safeguards varies with the intended use of the data communications medium. For short data transfers VRC is adequate. If the data channel is characterized by error bursts, it may be advisable to use a combination of VRC and LRC. For large transfers of data, the overhead of the foregoing methods cannot be tolerated if efficient use is to be obtained from the communication system. In these cases cyclic redundancy checks are generally used because of their ability to check large blocks of data using few check characters.

If the protocol is designed so that the receiver must interpret the received words to determine the nature of the transaction, the protocol is called character-oriented or byte-controlled. If the protocol is designed so that the structure of the message is determined without regard to the words transmitted, the protocol is called bit-oriented. In the latter case the name bit-oriented is used because the protocol uses specific bit patterns to delimit a fixed message structure. Users of the protocol must take care that the reserved pattern does not occur in the data to guarantee reliable transmission of data. Signetics data control chips are designed to accommodate both types of protocols. IBM's Binary Synchronous Communications Protocol (BiSync) is an example of a character-oriented protocol. High-level Data Link Control (HDLC) as defined by the International Standards Organization (ISO) is a bit-oriented protocol.

IMPLEMENTATION

In any data communications implementation, two major objectives are to minimize cost and to maximize reliability. The degree to which these objectives can be accomplished is a function of the means used to implement the objectives. The Signetics' 2652, 2653 and 2661 integrated data communications chips provide a compact, cost efficient, and flexible set of data communications building blocks.

These chips allow data communications systems designers to address requirements of their protocols whether in synchronous or asynchronous modes since the controllers are designed to take over much of the data communications requirements for interpreting the protocol rules under operating conditions. For example, the 2661, a universal synchronous/asynchronous data communications controller chip, provides special support for the Binary Synchronous Communications (Bisync or BSC) protocol which frees the host system from pre-processing data to ensure that idiosyncrasies of the protocol do not result in data loss.

The 2652 chip, which is dedicated to synchronous protocols, formats, transmits, and receives data for bit-oriented or byte-controlled protocols. This chip also includes support for Bisync operation. Transparency of synchronous communications is further enhanced for bit oriented protocols since the 2652 can recognize and create special characters such as message delimiters (flags), sync characters and other link control operators.

One of the most effective designs for reliability is the Signetics 2653 Polynomial Generator/Checker. This chip supports character-oriented protocols by creating and checking several types of message redundancy characters. Not all characters of the message may be subjected to the redundancy check, depending upon the protocol used. The 2653 specifically accommodates the protocol variations in this regard.

The Signetics 2652, 2653, and 2661 are all completely compatible to allow for the integrated implementation of a data communications network.

DATA COMM ASYNC AND SYNC TRANSMISSION

DATA COMMUNICATIONS ASYNC AND SYNC TRANSMISSION

ASYNCHRONOUS AND SYNCHRONOUS DATA COMMUNICATIONS

Data communications is the technology of transferring digital information from one device to another using phone lines, satellites or a combination of these and private communications lines. The main need for this type of information transfer arose when it became necessary for several computers to use the same data, for one computer to use data produced by another computer, and for a computer to transfer data between itself and its peripheral devices, such as terminals and printers. In essence, data communications is a method of passing, sharing, and disseminating information anywhere in the world via electronic technology. A simplified diagram of a data communications network is shown in Figure 1.

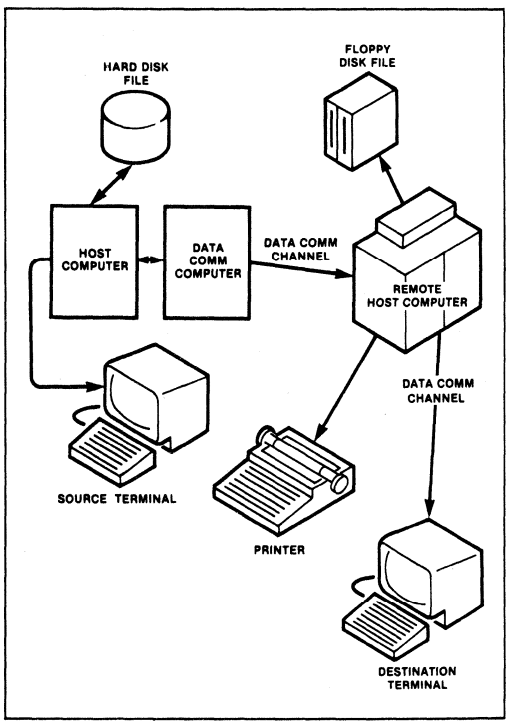


Figure 1.

DATA DEFINED

The smallest unit of data used by data processing machines is the binary digit, called the bit. A binary digit has two values, consequently two values of data can be represented by a bit. In order to deal with the large amounts of data typically communicated, groups of bits, called words, represent units of data. Thus, instead of being limited to the two values of a single bit, data processing machines can represent millions of data items by assigning a specific combination of bit values to a single data item.

For example, six bits are sufficient to represent 64 different characters. For this reason, some codes use six bits to represent the alphabet since 26 lower-case letters plus 26 capital letters ads to 52, leaving 12 combinations for the ten numerals 0 to 9. When it comes to specifying exactly where in a large manual a specific word is located, combinations of bits may be used to form addresses for the words in the document. Clearly, millions of different addresses would be required for large documents.

CLOCKS

Data processing machines usually process words rather than bits. All bits may be processed at once or they may be processed one bit at a time. If the process operates on a word, the process is called a parallel operation. If the process operates on a single bit at a time, the process is termed serial. The interval during which a bit or word is processed is called a basic (data processing) machine cycle. This interval is usually the period of a signal which continuously repeats. The repeating signal is called the clock for the data processing system.

DATA TRANSFER

Data can be transferred between parts of the same device or between two or more devices. As long as data transfers only occur from one part of a machine to another part of the same machine, the communications problems are minimal because the sending and receiving parts of the machine can be forced to be ready at the same time. Forcing such a situation is as simple as using the same clock to establish the basic processing intervals for the machines. The advantage of using the same clock to coordinate the data transfer is that the reliability of the data transfer is maximized because the data which the receiver records can be predicted to be the data that the transmitter sends, since the only variables in the transfer are predictable delays required for transmission.

Whenever the same clock is used to coordinate the transfer of data between data processing units, the transfer is called synchronous. If the data processing units use different clocks, the transfer is called asynchronous.

SERIAL TRANSMISSION

When the sending and receiving circuits are separated by a distance, from several feet to thousands of miles, it is not cost-effective to transfer data in the parallel form. Instead, serial transmission is used. Since serial operations process a single bit at a time, the cable required for a serial transfer only requires enough wires to accommodate a single bit. The usual form for such a serial channel is two wires which exhibit a difference in potential to represent the value of the bit processed.

Serial channels are more economical but less reliable than parallel channels. The reason for this loss of reliability is that cost constraints preclude addition of the cable to transfer the timing pulse that indicates when the data on the line is valid. Solutions to this problem without adding another pair of wires to the cable form the basic transmission disciplines of synchronous and asynchronous transmission techniques. If the solution includes transmission of the clock, it is called synchronous. If the solution excludes transmission of the clock, it is called asynchronous. All solutions involve a certain set of rules for how to interpret the information on the line at the receiving end of the channel. The set of rules is called a protocol.

ASYNCHRONOUS TRANSMISSION

Asynchronous transfers are widely used to allow communication between slow speed devices and a host computer. An example of such transfer occurs when keyboards and printers communicate with the host computer to which they are attached. In both instances the data is transferred at a rate which is slow in comparison to computer speeds. In addition, the number of bits transferred is relatively few so retransmission in case of errors costs little. These and other applications with similar data transfer rates are ideal for asynchronous transmission because the low data rates and the short transfers minimize the risk of mis-alignment of the time references of the sending and receiving machines.

In asynchronous transmission, the transmission medium, say two wires, is held in one state as long as data is not being transferred. When data is to be transferred, the transmitting machine signals the receiving machine that data is about to be sent by holding the line in the opposite state for a fixed length of time. After the specific interval, known to both ends of the line, the data is sent. After the message, the line is again held in its initial state for a specific time before a new message can be sent by the transmitter. The fixed intervals are measured in terms of the rate at which bits are changing at the input to the line. The initial interval is the start period and the final interval is the stop interval. These periods delimit the start and end of a word transfer. For this reason, this form of asynchronous transfer is called START-STOP transmission. The lengths of the intervals will vary from one application to another based upon the time required for the receiver to initialize itself at the beginning of a message transfer and the time required to return to its initial state at the end of a message transfer.

SYNCHRONOUS TRANSMISSION

Synchronous transmission is normally used where rate of transfer of data is high. An example of such high-speed data transfer occurs when computers must communicate or when a computer must communicate with an intelligent peripheral such as a buffered printer or word processing station. In these and similar applications, large amounts of data must be transferred in short intervals. In such applications it is inefficient to add extra start and stop intervals to each word transferred because that time could be used to transfer data bits instead.

Asynchronous techniques work because the receiver can guess reasonably accurately when each bit will arrive if it knows when the word begins. In the case of synchronous transmission data speeds, the lack of start and stop bits prevents the receiver from guessing accurately enough to allow for data to be recorded without a clock. The solution therefore is to send the clock along with the message. In some cases, the clock accompanies the data via a separate pair of wires. In other cases, the clock travels along the same pair of wires as the data through the use of special data representation schemes called self-clocking encodings.

While it is possible for synchronous techniques to dispense with the time interval required for the start and stop bits of the characters, it is not possible to dispense with the character separation function of these delimiters. Thus synchronous transmission schemes need to provide a means of indicating the specific bit at which a character begins and ends. In the case of asynchronous techniques, this was indicated by the end of the start and stop intervals. In the case of synchronous transmission schemes, the receiver is synchronized to the first character of the message using special characters. Subsequent characters are assumed to follow the initial character until the occurrence of a specified limit condition such as a count of the characters received or the receipt of another special character. The transmitted clock ensures that no characters are missed during the transfer, barring interference from outside agents.

IMPLEMENTATIONS

The Signetics 2661 and 2652 are controller chips that allow data communications systems designers to implement solutions to their communications needs in either asynchronous or synchronous formats. The 2661,

a programmable communications interface, operates in either synchronous or asynchronous modes. Since the chip interfaces directly to most eight-bit processors, the user may program, via software, the controller to handle either format. If operating in the asynchronous mode, the user may select from three stop bit options and a character length varying from five to eight bits. Assuming that the systems designer is using a standard synchronous protocol, say Bisync, the 2661 will unburden his host processor by detecting and inserting special control characters required for the protocol.

In synchronous data communications systems, the Signetics 2652 formats, transmits and receives data compatible with five major communications protocols. This monolithic communications controller handles bit oriented protocols as well as word oriented protocols, performing such functions as detection of special protocol control words such as SYNC, FLAG, GA. These characters must be interpreted at some point in data communications systems. By handling the interpretation of the general characters, the 2652 allows the host processor to be more free to handle higher level tasks. In bit-oriented protocols, the 2652 transforms the data to ensure that no data words have the appearance of message delimiters. This feature, called bit stuffing, is especially important since every character must be checked and changed if it looks like a control flag. At the receiving end, the 2652 removes the extra bits to ensure that the received data agrees with the transmitted data.

For asynchronous format requirements, the Signetics 2681 provides two independent receiver/transmitter pairs in a single package. Included are bit rate generators for popular communications speeds, fully programmable data formats, and a counter/timer circuit for auxiliary use.

DATA COMM ERROR CONTROL

DATA COMMUNICATIONS ERROR CONTROL

PROTOCOLS

Whenever information is transferred from one physical location to another, it is often the case that data arriving at the intended destination differs from the data sent. This is an unavoidable consequence of information transfer. Since some of the data arrives without modification some of the time, the problem is one of probabilities. Error control is often a matter of adding clues to messages to allow the receiver to answer the question: Is the data at the receiver the same as the data that left the transmitter?

If the receiver has perfect information concerning the transfer, the entire message can be reliably reconstructed at the receiver. In such a case, the penalties of errors during transmission can be avoided. These cases are covered in the discipline of error correction. In the absence of perfect knowledge, the clues in the transmitted message might be sufficient to determine that the data received is not the same as the data transmitted. This is embodied in the error detection discipline. Error detection is the foundation on which error correction is based.

In data communications, error control is practiced at the physical level and at the message level. The physical level includes the interface between the data communications network and the devices using the network for transfer of data. The physical level also includes the physical representation of the data as it is being transferred through the communications network. The message level includes the various protocols used to transfer blocks of data from one point in the network to another.

The amount of error control required for a data communications network is often a function of the amount of error correcting capability of the receiver. For example, humans often detect errors based upon the context of the information transferred. If the word "context" had been spelled "cintext", a human reader would have correctly guessed that the data intended should have been "context". In some applications, it is possible to take advantage of the fact that a human is in the loop. This mode of error control is sometimes acceptable in low-speed terminal communications. The method of control is to effect a rule (protocol) in which the receiving machine transmits back to the terminal every character which the terminal sends to the host. When the character arrives back at the terminal, the human verifies that no error occurred during transmission by comparing what appears on the screen or paper to what was entered (Figure 1). Although there are some flaws in this method, they are easily overcome by retransmission and the cost of the system is minimal.

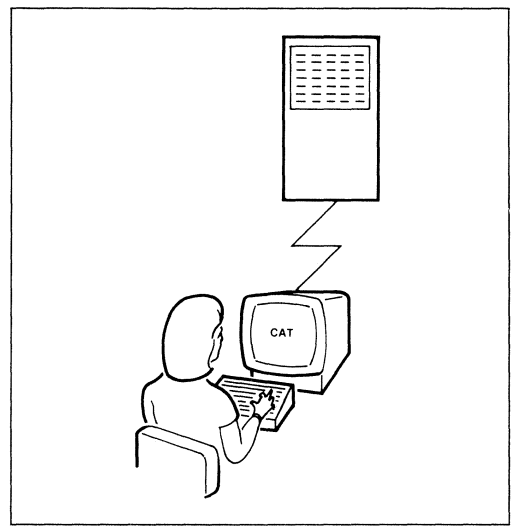


Figure 1. Human Error Detection

SYSTEMS INTEGRITY

In cases where ignoring data communications errors does not give acceptable data communications performance, the wise approach is to include the least amount of control consistent with effective communications. The reason for the minimization guideline is that error control improves in proportion to the amount of capital

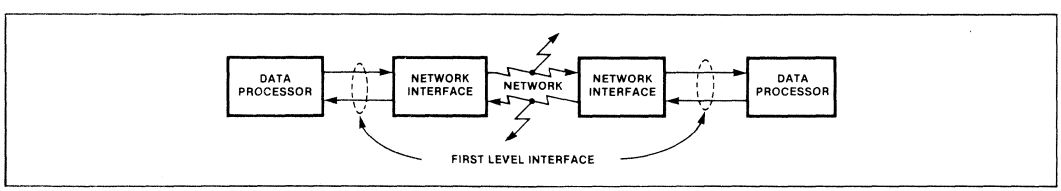


Figure 2. Interface to the Data Communications Network Should be Secure

invested and the amount of electronics dedicated to the task. Since all error control is probabilistic (i.e., you can't catch every error), increasing levels of protection come at a price that is disproportional to the added coverage.

Violation of this law of diminishing returns is often observed at the interface between the data communications network and the systems it serves. The first line of error control exists at the interface between the data communications network and the using devices. Yet, it is often the case that reliable data transmission is requested within a data communications system but no care is taken to ensure that the data offered to the network is free of error (Figure 2). If the data transferred features a parity error at the input to the system, the receiving device will receive incorrect data, faithfully transferred by the communications network, in the case of an error-free transmission.

It is often a simple matter to add a parity check to data within the source/sink devices to ensure data integrity on a systems basis. In this manner, needless error recovery procedures will not be activated to isolate network errors that do not exist.

The next level of error control is to design data transfer protocols which guarantee that the messages transferred by the system contain enough redundancy to ensure that errors are detected and recovery procedures begun (Figure 3). In many cases, the cause of the error is of a temporary nature. Consequently, retransmission is sufficient to recover since the disturbing phenomenon will have ceased. When multiple messages must be transmitted, each message should have a system assigned number to ensure that errors can be detected early and that retransmission can be limited to the erroneous portions. Modern protocols, such as SDLC and DDCMP, feature message numbers to allow for the detection of loss of messages and the loss of the sequences of messages. BiSync, the original synchronous protocol developed by IBM in the 1960's features an optional header field in which such information might be incorporated.

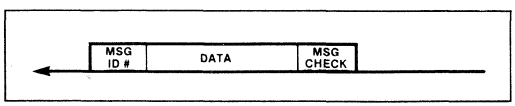


Figure 3. Reliability Incorporated in Message Structure

ERROR CHECKING AND RECOVERY

An important function of a line protocol is to assure correct reception of data. Communications facilities are error-prone. To compensate for this, line control procedures include the generation, transmission, and testing of check bits. These check bits (often called Block Check Characters—BCC) make up the trailer field *of the transmission block.* They are generated by a checking algorithm which is usually applied only to the information field of a block.

Each block of data transmitted is error-checked at the receiving station in one of several ways, depending on the code and the functions employed. These checking methods are Vertical Redundancy Checking (VRC), which is parity checking by character as the data is received; and either Longitudinal Redundancy Checking (LRC) or Cyclic Redundancy Checking (CRC), which check the block after it is received. *After each transmission, the receiving station normally replies with a positive (ACK) acknowledgment (data accepted, continue sending), or with a negative (NAK) acknowledgment (data not accepted; e.g., a transmission error was detected—retransmit previous block). The acknowledgment may be in a special control message (ACK) or in the response field in the header of the next message to be send in the opposite direction.* Retransmission of a block of data following an initial NAK is usually attempted a number of times. Until the block is accepted, the data buffer cannot be released by the transmitting system.

VRC (Vertical Redundancy Checking) is an odd or even parity check performed on a per-character basis and requires a parity check bit position in each character. If individual characters are represented by eight bits, such as when using an eight-level code, seven may be used to represent actual numbers and letters, and the eighth may be reserved for checking purposes. The presence or absence of the eighth bit provides the inherent checking feature. For example, in an even parity check, the parity bit is used to make the total number of one bits in the character even. If the character contains four zeros and three ones, then a one bit is inserted as the parity bit.

Longitudinal Redundancy Checking (LRC) is a technique for checking an entire message or block of data. In this case, an exclusive "OR" logic is used for all the bits in the message and the resulting character, called the *Block Check Character (BCC),* is transmitted as the last character in the block. The receiving device independently performs the same counting procedure and generates a Block Check Character. It then compares its own BCC character with the one received. If they are not identical, an error condition exists, and the sending device is notified that an error condition exists within the block. *LRC is frequently used in conjunction with VRC to increase the error detection capability within a system.*

Cyclic Redundancy Checking (CRC) is a more sophisticated method of block checking than LRC. *This type of error checking involves a polynomial division of the data stream by a CRC polynomial. The 1's and 0's of the data become the coefficients of the dividend polynomial while the CRC polynomial is preset. The division uses subtraction modulo 2 (no carries) and the remainder serves as the Cycle Redundancy Check.* The receiving

station compares the transmitted remainder with its own computed remainder, and an equal condition indicates that no error has occurred.

There are many constants that may be used to perform the CRC division. Two of the most popular versions are called *CRC-16* (which uses a polynomial of the form $x^{16}+x^{15}+x^2+1$) and *CRC-CCITT* (which uses a polynomial of the form $x^{16}+x^{12}+x^5+1$). Each generates a 16-bit BCC. CCITT, the International Consultative Committee for Telephony and Telegraphy, is responsible for usage standards.

Error checking also involves checking for sequence errors. Protocols handle this in different ways. These include alternating acknowledgments and block sequencing. The technique used depends on the protocol. The receiving station sends back an indication of a sequence error with a negative acknowledgment or some other control message.

IMPLEMENTATION

Signetics offers a comprehensive set of error control features in its data communications controller chips, the 2652, the 2653 and the 2661. This chip set is completely compatible, allowing the designers of data communications systems the ability to implement error control at the system level and at the protocol level. The Signetics 2653 Polynomial Generator/Checker supports block check coded error detection and parity generation and checking, featuring CRC-16 and CRC-12 which are able to detect 99 percent of the burst errors occurring up to the length of the code used. Since the 2653 operates in the parallel mode, it can accomplish parity generation and checking on the data bus of the system using the data communications network, thereby providing the first line of protection for systems level data integrity, as indicated in Figure 2.

Within the data communications network, the entire family of Data Communications Controllers support the latest protocols and feature special functions which allow for the accurate transmission of messages formatted according to both bit-oriented protocols and character-oriented protocols. Violations of the message structure are detected by special functions built into the chips to account for automatic detection and checking of the block check characters in BiSync messages (Signetics 2661). Message delimiting and detection is automatically accomplished by the programmable features of the Signetics 2652 which provides automatic bit-stuffing and deletion to ensure that transmitted messages adhere to bit level protocol. Special control patterns, Flag, Go Ahead, and Abort, are generated and detected by the Signetics 2652 as it automatically monitors the link level control characters of SDLC and similar advanced protocols.

The Signetics 2652 and the Signetics 2661 both feature a maintenance mode of operation which allows for isolation of errors on a node basis. This mode of system verification is extremely important for cases in which it is necessary to determine whether the receiver is at fault, the transmitter is at fault or the channel is at fault. This method of system verification is called Loop-back. In operation, the output of the transmitter is directed to the receive circuitry (Figure 4) so that the transmitting station can verify that no errors appear in the data it is sending.

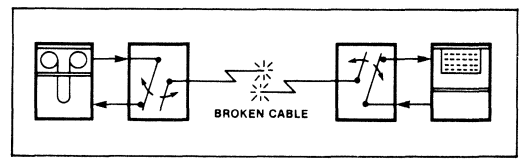


Figure 4. Detecting a Channel Fault in Maintenance Mode

BINARY SYNCHRONOUS COMMUNICATIONS

BINARY SYNCHRONOUS COMMUNICATIONS

PROTOCOLS

Communications among data processing machines invariably become necessary when one machine produces information and another processes it. The rules governing the transfer of information from one point in a data communications network to another are called protocols. While there are many kinds of data communications protocols, there are two major categories of synchronous protocols—character oriented protocols and bit oriented protocols. The distinction between these types of communications protocols lies in the manner in which the various portions of a message are delimited.

Character oriented protocols use a partitioned alphabet (Figure 1). One part of the alphabet is used to write messages to the data communications network. The other part is used to write the data messages transferred by the network. The special part only used to write messages to the network elements is called the control set or control characters. The part of the alphabet used to write data messages is called the graphic set, or graphic characters.

One of the most widely used character oriented protocols in data communications is the Binary Synchronous Communications Protocol (Bisync or BSC). This protocol was designed by the International Business Machines company for use in their terminal products in the early 1960's.

To use the protocol, special send/receive devices were constructed that took the appropriate actions when the outgoing message required the insertion of a communications link control character or when the incoming information contained a character from the control set that indicated special action.

As long as the information portion of the message only used characters from the graphic set, Bisync worked well. However, there were cases in which the information that was transferred might violate the groundrules and include graphic information which actually looked like a control character. For example, suppose that the information for which transfer is desired happened to be the patterns of one and zeros formed by the digitization of a picture. This could easily occur if the information arose from a half-tone plot or a facsimile transmission. In this case, the pattern of light and dark points could create a series of ones and zeros that coincide with the pattern of the control character.

In order to circumvent the possibility of improper operation in the controllers designed to interpret the control characters, the graphic set included a special character, called a data link escape (DLE) character. Whenever the controller detected this character followed by an 'STX' control character in the incoming data stream, it would ignore control characters in the incoming data until the next data link escape character. Reception of a single data link escape character followed by a legitimate character from the control set indicates the transmission of a control character by the sending controller. Reception of a control character not preceded by a single data link escape character merely indicates the transmission of information. This sub-mode of the BSC protocol is called the transparent mode.

Data Link protocols perform the following data communications activities:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.
5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

Each of the foregoing data communications activities must be specified by one or more of the characters in the BSC control set. In some cases a single character will effect the function. In others, a function will be defined by a sequence of transmissions.

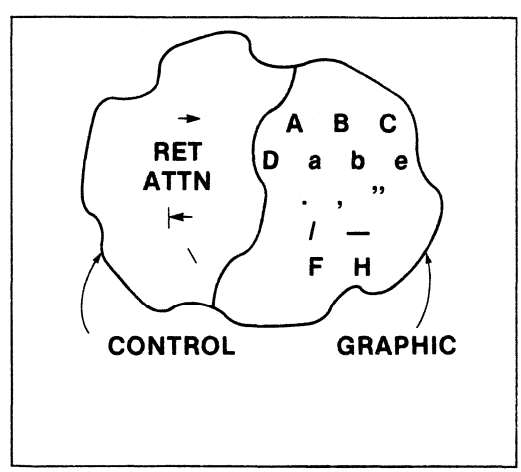


Figure 1. Partitioned Alphabet

OPERATING MODES

To grant the sender access to the physical channel, the Binary Synchronous Communications protocol features two modes of transmission. The non-transparent mode of transmission is the mode of transmission in which all information transferred only uses characters from the graphic set. The transparent mode is the case in which the graphic information is allowed to contain characters from the control set.

FORMAT

Protocol activity is often graphically represented as a plot of data transfer as a function of time. Since the data and control information usually takes the form of words, the plot is usually a sequence of horizontal bars that represents the type of information on the communication channel at a given time. In the case of synchronous communications, the blocks may be of various lengths since several words may be transmitted without a break in the transmission. If the protocol is asynchronous, the plot will consist of a series of discrete word plots, sometimes with the start and stop bits indicated on the plot.

The plot below (Figure 2) depicts message sequencing using the Binary Synchronous Communications Protocol. It is apparent that no messages are transmitted at the same time messages are being received. This characteristic of Bisync is called half-duplex operation. In this mode of operation, only one station can use the communications channel at a time. For this reason, the plot shows the received messages falling in the time intervals for which there are no transmit messages.

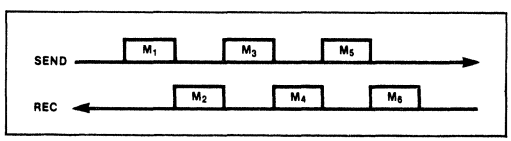


Figure 2. Message Sequencing

CONTROL SET

Every message transferred according to the BSC protocol is controlled using one or more of the special characters reserved for controlling the transfer of messages. Bisync uses fifteen control characters to effect orderly transfer of information. The binary code used to represent the codes varies according to the data format used, i.e., ASCII, EBCDIC, etc. Regardless of the code used, unique characters are reserved for the following functions:

ACK0	Acknowledgment of even numbered transactions
ACK1	Acknowledgment of odd numbered messages
ARQ	Automatic Request for retransmission
DLE	Data Link Escape
ENQ	Enquiry
EOT	End of Transmission
ETB	End of Transmission Block
ETX	End of Text
ITB	Intermediate Text Block
NAK	Negative Acknowledge
RVI	Reverse Interrupt
SOH	Start of Header
STX	Start of Text
SYN	Synchronize
TTD	Temporary Text Delay
WACK	Wait for Acknowledgment

The BSC message format is shown in Figure 3. Different portions of the message are delimited by control characters which signal the receiver of the next action to expect or perform. The header portion of the message contains user defined information. Some of this information may be address and status data. In Bisync, the header is an optional feature. It is not necessary that every correct implementation of Bisync include a header.

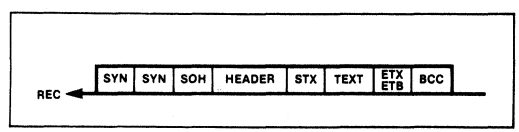


Figure 3. Message Format

A message session begins with the sending station issuing a request for a session by transmission of an ENQ character (Figure 4). If the receiving station is not ready to receive, it responds with a NAK character. If the sending station continually requests a communications session, when the reciever is ready to accept, it responds with an ACK character. The transmitting station subsequently begins with the transfer of messages in the format shown in Figure 3. The session ends with the transfer of an EOT character.

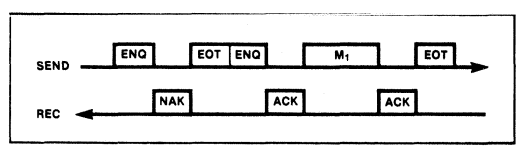


Figure 4. Sample Message

When the message contains information that may be mistaken as control characters by the receiving controller, BSC offers the transparent mode of transmission. The transparent mode of data transfer is the mode in which the receiver ignores all characters until notified

to resume observing the control characters. This mode of transfer begins when the transmitter prefixes a DLE character to the STX character at the beginning of the text portion of the message (Figure 5). When the receiver detects the DLE-STX, it ignores the remaining control characters of the message until a message delimiter occurs which is preceded by a DLE character.

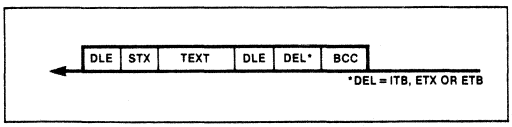


Figure 5. Transparent Format

Since the DLE character is the means of entering and exiting the transparent mode of communication, special care must be taken to ensure that DLE characters occurring in the data do not cause premature exit from the transparent mode. The technique for accomplishing this is to pre-process all the data and insert DLE characters in front of each of the data characters having the same format as the DLE character. The receiver strips the inserted DLE and stores the original DLE.

At the end of the information transfer, the ending delimiter is inserted as usual with the exception that the delimiter is preceded by a DLE character. When the receiver detects the delimiter preceded by the DLE, it resumes control character interpretation, treating the next two characters as the block check character (BCC).

MESSAGE INTEGRITY

To assure integrity of the data transfer, all Bisync text messages append a two-character block check character (BCC) field at the end of the message. The BCC is derived by treating the message as a polynomial, dividing it by a fixed polynomial, and using the remainder as the BCC.

In operation, the transmitter appends the BCC to the message. The receiver regenerates the BCC and compares it to the received BCC. If they match, an ACK0 or ACK1 is sent back and the transmitting station proceeds with the next message. If the BCCs don't match, the receiver returns a NAK and the transmitter re-sends the last message. If more than a predetermined number of NAKs are received, the transmitting station assumes a faulty condition and alerts the operating system for appropriate action.

The first SOH or STX to follow a line turnaround resets the BCC logic. All succeeding STXs or SOHs until the next line turnaround are included in the block checking. ITB, ETB, and ETX are always included in the BCC and are followed by the block check characters associated with the portion of the message in which they occur. In the transparent mode of transmission, false ITB, ETB and ETX characters are ignored since they are not preceded by DLE. The actual message terminator characters are preceded by DLE in the transparent modes. The actual terminators are included in the BCC calculations in the transparent mode but the extra DLEs inserted to ensure transparent transmission are excluded. SYN characters used as line fill are not included in the BCC.

IMPLEMENTATION

The Signetics 2661, a universal asynchronous/synchronous data communications controller chip, offers special support for data communications systems implemented using BSC as the link control discipline. The chip provides automatic SYN or DLE-SYN insertion and stripping, eliminating a major preprocessing load from the host system. Both transparent and normal modes of transmission are accommodated by the chip. Since Bisync may be implemented with several character sets, the Signetics 2661 operates with equal facility on character lengths ranging from five bits to eight bits.

The Signetics 2653 Polynomial Generator/Checker is a polynomial generator and checker which provides character level checking and message checking. The Binary Synchronous Communications protocol may be used with parity checking on each character or a group of characters, or it may be used with the CRC-12 or CRC-16 polynomials for cyclic redundancy checking. The Signetics 2653 implements both types of parity checking and both cyclic redundancy check polynomials. This chip is fully compatible to the normal and transparent modes of Binary Synchronous Communications protocol operation, and automatically excludes or includes characters in the accumulation as required by the protocol. Additionally, the 2653 can detect the receipt of control characters and signify the controlling microprocessor of their occurrence.

DIGITAL DATA COMM MESSAGE PROTOCOL

DIGITAL DATA COMMUNICATIONS MESSAGE PROTOCOL

PROTOCOLS

Communications among data processing machines invariably become necessary when one machine produces information and another processes it. The rules governing the transfer of information from one point in a data communications network to another are called protocols. While there are many kinds of data communications protocols, there are two major categories of synchronous protocols—character oriented protocols and bit oriented protocols. The distinction between these types of communications protocols lies in the manner in which the various portions of a message are delimited.

Character oriented protocols use a partitioned alphabet (Figure 1). One part of the alphabet is used to write messages to the data communications network. The other part is used to write the data messages transferred by the network. The special part only used to write messages to the network elements is called the control set or control characters. The part of the alphabet used to write data messages is called the graphic set, or graphic characters.

One of the character oriented protocols used in data communications is the DEC Digital Data Communications Message Protocol (DDCMP) which was designed by the Digital Equipment Corporation for use in their terminal products. DDCMP is character oriented because it uses special characters which determine the nature of the communications transaction. All transactions are based upon the exchange of message sequences instead of unique characters. All special characters are embedded in the message structure. The message format separates the control messages from the graphic information. This separation provides 'transparency', that is, it allows the transmission of data patterns that have the same bit structure as the control set without violating the integrity of the protocol or the information.

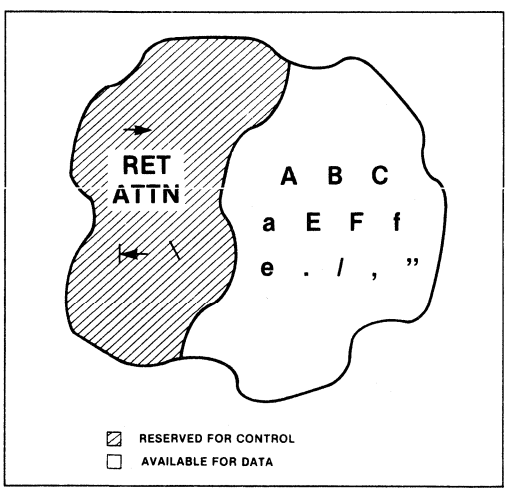


Figure 1. Partitioned Alphabet

The ability to transmit data which has the appearance of control is important to the design of a data communications network because there are cases in which the information transferred might actually look like a control character. For example, suppose that the information for which transfer is desired happened to be the patterns of ones and zeros formed by the digitization of a picture. This could easily occur if the information arose from a half-tone plot or a facsimile transmission. In this case, the pattern of light and dark points could create a series of ones and zeros that coincide with the pattern of a control character, causing an improperly designed protocol to assume unpredictable states.

To use the protocol, special send/receive devices are constructed that take appropriate actions when the outgoing message requires a communications link control character or when the incoming information contains a character from the control set.

Protocols perform the following data communications activities:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.
5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

Each of the foregoing data communications activities must be specified by one or more of the control messages in the DDCMP control set. In some cases a single message will effect the required function. In others, a function will be determined by a sequence of messages defined by successive transmissions.

OPERATING MODES

The DDCMP communications protocol features three modes of transmission. The control mode of transmission is used to control the data communications facilities. The information mode is used for transfer of information by the data communications devices using the system. The maintenance mode is used to initialize network controllers in what is called down-line loading.

Down-line loading is a technique for transferring an operating program to a device which does not feature permanent storage of its operating programs.

FORMAT

Protocol activity is often graphically represented as a plot of data transfer as a function of time. Since the data and control information usually takes the form of words, the plot is usually a sequence of horizontal bars that represents the type of information on the communication channel at a given time. In the case of synchronous communications, the blocks may be of various lengths since several words may be transmitted without a break in the transmission. If the protocol is asynchronous, the plot will consist of a series of discrete word plots, sometimes with the start and stop bits indicated on the plot.

The plots of Figure 2 illustrate the message flow resulting when using the DEC Digital Communications Message Protocol. The distinguishing feature of the plot is that the messages in one direction do not necessarily fall in the blank intervals defined by messages going in the opposite direction. This indicates that message M_i is not necessarily related to message M_j. The overlapping of messages on the time plot indicates that the protocol can be used in full-duplex mode and that the time normally required to reverse direction on a half-duplex line can be used for productive communications. However, the protocol also allows for half-duplex (alternating) operation if desired.

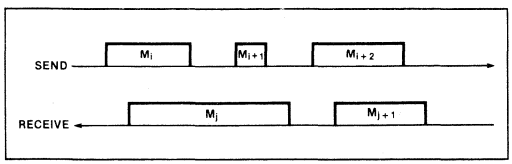


Figure 2. Message Flow

CONTROL SET

Figure 3 presents the basic DDCMP message structure. Note that the message consists of a header and an information portion. The information portion is optional but every legitimate message must contain a header. The binary code used to represent the characters is ASCII, using even parity. The SYNC bytes are eight bit characters which are used to allow the receiving circuitry to detect character boundaries.

The TYPE field contains one of the control characters to indicate what type of message is being sent. Data messages are indicated by use of the character SOH, start of header. The control messages are indicated by the character ENQ, enquiry. Maintenance messages are indicated by the character DLE, Data Link Escape. If the message is a control message, it is either an acknowledgement, ACK, or negative acknowledgement, NAK.

The Operand field consists of two sub-fields, a 14 bit MODIFIER field and a 2 bit FLAG field. If data is being transferred, the MODIFIER contains a count of the number of bytes in the information field, including the CRC bytes. This count provides transparency, since the actual bytes in the information field can be of any type. If the message is a control message, the MODIFIER clarifies the type of ACK or NAK indicated by the TYPE. There is only one type of ACK. Several types of NAK's are indicated by the MODIFIER. For example, a NAK may be due to a buffer overrun condition or a block check error on a preceding message.

The high order bit of the FLAG field flags the occurrence of a SYNC character at the end of the current message to allow the receiver to re-initialize its synchronization detection logic. The low order bit of the FLAG field indicates the current message to be the last of a series the transmitter intends to send. This allows the addressed station to begin transmission at the end of the current message.

The RESPONSE and SEQUENCE fields contain message numbers. Every station assigns a sequence number to every message it transmits. The station-assigned sequence number is placed in the SEQUENCE field. If message sequencing is lost, the control station can request the number of the last message transmitted by a certain station. When the request is received, the answering station places the last assigned sequence number in the RESPONSE field.

In a multipoint configuration, Figure 4, stations intended to receive a specified message are indicated by the entry in the ADDRESS field. The CRC 1 allows for detection of errors in the header portion of the message.

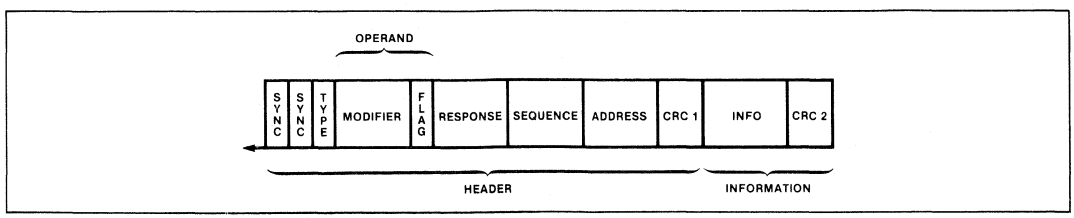


Figure 3. Message Format

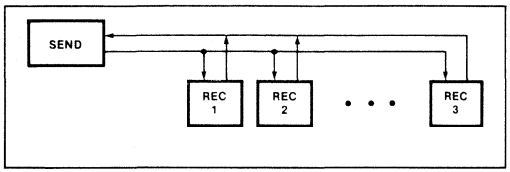


Figure 4. Multi-Point Network

The information field may contain any information the transmitting device provides, including special control characters. Proper transmission and proper system operation is assured by the DDCMP MODIFIER field which contains a count of the number of data bytes in the information field. The receiving controller loads the count and allows the appropriate number of bytes to pass before resuming active participation in the data link control. At the end of the data count, the error detection logic can be activated to check the CRC 2 field.

ERROR DETECTION AND RECOVERY

To assure integrity of the message transfer, the DDCMP messages append a two-character cyclic redundancy check (CRC) field at the end of the header field and at the end of the information field. The CRC is derived by treating the header or information field as a polynomial, dividing it by a fixed polynomial, and using the remainder as the CRC.

In operation, the transmitter appends the CRCs to the message. The receiver re-generates the CRCs and compares them to the received CRCs to determine if an error has occurred.

When an error occurs, DDCMP sends a separate negative acknowledgment (NAK) message. DDCMP does not require an acknowledgment message for all messages. Only when a transmission error occurs or if traffic in the opposite direction is light (no data message to send) is it necessary to send a special NAK or ACK message, respectively. The number in the response field of a normal header or in either the special NAK or positive acknowledgment (ACK) message specifies the sequence number of the last good message received. For example, if messages 4, 5, and 6 have been received since the last time an acknowledgment was sent and message 6 is bad, the NAK message specifies number 5 which says "message 4 and 5 are good and 6 is bad." When DDCMP operates in the full-duplex mode, the line does not have to be turned around. The NAK is simply added to the sequence of messages for the transmitter.

When a sequence error occurs in DDCMP, the receiving station does not respond to the message. The transmitting station detects from the response field of the messages it receives (or via timeout) that the receiving station is still looking for a certain message and sends it again. For example, if the next message the receiver expects to see is 5 and it receives 6, it will not change the response field of its data messages which contains a 4. This says: "I accept all messages up through message 4 and I'm still looking for message 5."

OPERATION

In operation, the transmitting station sends a Start control message to the station indicated in the Address field. The addressed station responds with a Start ACK (start acknowledge) message. The transmitting station subsequently formats and transmits successive information messages until the transaction is complete. If the receiving station is transmitting at the same time, the original transmitter will know the status of the transfer by interrogation of the Response fields in the messages directed to it. Otherwise, a special control command, Reply, may be issued to the receiving station. In this case, the receiving station will respond with a control message that indicates the last message properly received.

IMPLEMENTATION

The Signetics 2652, a multi-protocol communications controller chip, offers special support for data communications systems implemented using DDCMP as the link control discipline. The chip provides automatic SYN insertion, detection, and stripping and internal CRC generation and checking, eliminating these processing requirements from the host CPU. The 2652 operates at data rates of up to 2 Mbits per second and is also capable of supporting other synchronous character oriented protocols and bit oriented protocols such as SDLC.

When asynchronous format needs exist, the DDCMP protocol interface function can be met by use of the Signetics 2661. This Programmable Communications Interface (PCI) supports both asynchronous and synchronous line formats. It features an internal baud rate generator with 16 commonly used communications frequencies. In the synchronous mode, DDCMP can be supported with automatic SYNC generation, detection, and stripping. The CRC function is implemented with a Signetics 2653 Polynomial Generator and Checker (PGC). The 2661/2653 combination can also be utilized to be used for the IBM bisync (BSC) protocol and its derivatives.

SYNCHRONOUS DATA LINK CONTROL (SDLC)

SYNCHRONOUS DATA LINK CONTROL (SDLC)

PROTOCOLS

Communications among data processing machines invariably become necessary when one machine produces information and another processes it. The rules governing the transfer of information from one point in a data communications network to another are called protocols. While there are many kinds of data communications protocols, there are two major categories of synchronous protocols—character oriented protocols and bit oriented protocols. The distinction between these types of communications protocols lies in the manner in which the various portions of a message are delimited.

Character oriented protocols use a partitioned alphabet (Figure 1). One part of the alphabet is used to write messages to the data communications network. The other part is used to write the data messages transferred by the network. The special part only used to write messages to the networks elements is called the control set or control characters. The part of the alphabet used to write data messages is called the graphic set, or graphic characters.

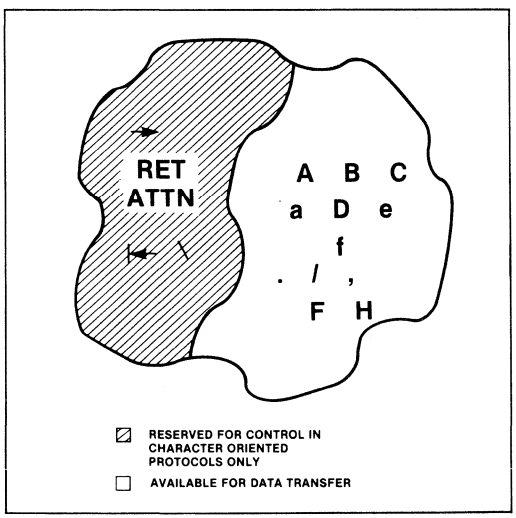


Figure 1. Transmission Alphabet

Bit-oriented protocols process information at the bit level rather than the character level to delimit message format. No characters are reserved for the control set, and only one bit pattern, 01111110, called the FLAG, is reserved. The FLAG delimits the beginning and end of a frame. Within the frame, positional significance is used to identify fields of the frame. All characters may appear at any part of the transmitted message without causing disruption of the link control hardware. In addition, since the transmission control is implemented at the bit level, there is absolutely no dependence upon the word width used. While some protocols require characters of a specified length, bit-oriented protocols are equally efficient regardless of the character length. A further advantage of bit-oriented protocols is that they cleanly separate the control of the transmission path (link) from the control of the devices using the transmission path. This latter advantage arises from the fact that control characters are used by devices while the link uses bits.

One of the most widely used bit-oriented protocols in data communications is the Synchronous Data Link Control Protocol. This protocol was designed by the International Business Machines company for use in their terminal products. The protocol serves the needs of products that must communicate in a network structure but do not feature the same character sets or lengths. In separating the link control from the device control, the protocol also makes it possible to create a layered telecommunications system which may be updated in modules as the need arises without affecting portions of the system which do not require changes in functions.

As stated above, the only special bit sequence is the FLAG, which is used to identify the beginning and end of the frame. It is possible that when data is transmitted a concatenation of characters may result in the presence of a FLAG character in the bit stream which would be incorrectly identified at the receiver as an end-of-frame FLAG. For example, an ASCII 'y' followed by 'c' would result in the serial bit stream '10011111100011' (LSBs transmitted first) which contains a FLAG.

In order to circumvent the possibility of improper operation, SDLC controllers are designed to postprocess all data prior to transmission. Whenever the controller detects a sequence in the data stream that consists of six consecutive ones, an extra zero is inserted after the first five ones to ensure that all transmitted data consists of streams of information that contains at most five consecutive ones (Figure 2). At the receiving end, the serial data is preprocessed to remove a zero which follows five consecutive ones. Thus, reception of a sequence consisting of more than five consecutive ones, subsequent to the preprocessor, signals an error condition or a control (FLAG) sequence.

Protocols perform the following data communications activities:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.

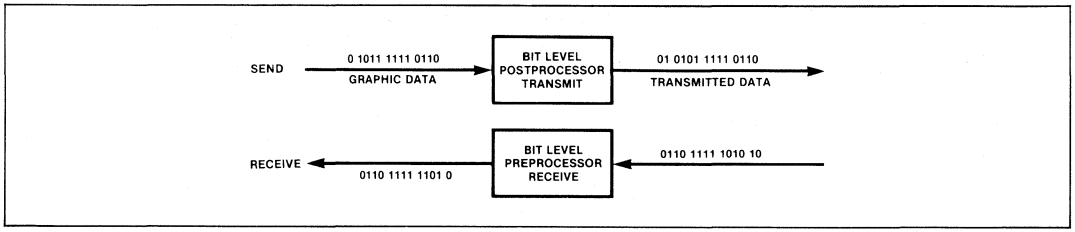


Figure 2. Bit Level Processing

5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

Each of the foregoing data communications activities is accommodated at the link level in SDLC. Path specification occurs at a higher level in the communications system. Specific characters effect communications functions only to the extent that the link must be controlled. Other communications functions are defined in the message structure of the Synchronous Data Link Control protocol.

FORMAT

Protocol activity is often graphically represented as a plot of data transfer as a function of time. Since the data and control information usually takes the form of words, the plot is usually a sequence of horizontal bars that represents the type of information on the communication channel at a given time. In the case of synchronous communications, the blocks may be of various lengths since several words may be transmitted without a break in the transmission. If the protocol is asynchronous, the plot will consist of a series of discrete word plots, sometimes with the start and stop bits indicated on the plot.

The plot in figure 3 shows the message flow occurring using the Synchronous Data Link Communications Protocol. It is significnt to note that transmitted messages and received messages may be enroute at the same time. This indicates that the SDLC protocol supports two-way communication on separate channels. This mode of communications is called full-duplex operation. However, the protocol also allows for half-duplex (alternating) operation is desired. Another feature to notice is that the incoming and outgoing messages have message numbers attached. In SDLC a receiving station may allow up to seven messages to be outstanding before acknowledgement. This mode of operation allows large blocks to be transmitted without waiting for a response from the receiver. Message M_i is not necessarily related to message M_j.

MESSAGE FORMAT

In granting access to the physical channel, SDLC features three modes of transmission. Supervisory mode is used for controlling aspects of the data link. Information transfer mode is used to effect the transfer of user data. Non-Sequenced mode is used to initialize and configure the communications system.

The SDLC message format is shown in Figure 4. Different portions of the message are determined by a character's positioning with respect to the Flag which signals the receiver that a message is ending or beginning. The header portion of the message consists of the first two bytes following the flag and the message block check character consists of the final two bytes of the message preceding the ending flag.

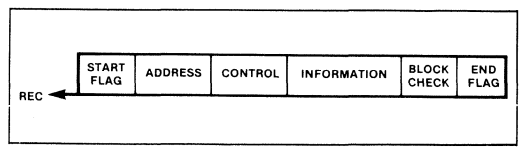


Figure 4. Message Format

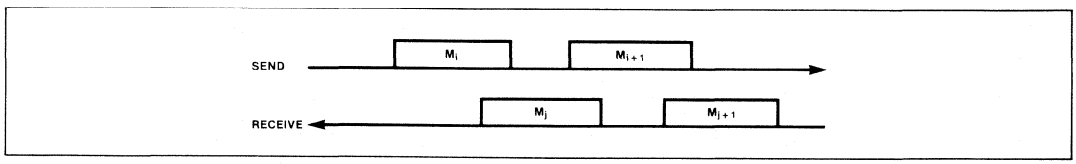


Figure 3. Message Flow

ADDRESS AND BLOCK CHECK FIELDS

The Address field identifies the station intended to respond to or receive the message.

INFORMATION FIELD

The Information field contains data to be transferred by the message. The Information field is optional. The Block Check field is a two byte cyclic redundancy check field.

CONTROL FIELD

The first three bits of the control field (Figure 5) specify a receive count for a supervisory frame or an information transfer frame. In a non-sequenced transfer, they simply modify the transaction. The fourth bit of all control fields is an indication of whether the current message is the last of a series of messages and whether the receiver is allowed to transmit data or required to respond to control messages.

The next three bits of the control field specify the number of messages received for an Information frame. For a Supervisory or Non-sequenced frame, the next two bits form an operation identifier field. The remaining bits of the control field are reserved. In the case of the Information frame, the bit following the message received count must be zero. The two bits following the two bit operation field in the Supervisory and Non-sequenced frames must be 01 to indicate a supervisory frame and 11 to indicate a Non-sequenced frame.

The operational identifiers of a supervisory frame are used to acknowledge transmissions, and request retransmissions. Pattern 00, called Receive Ready, indicates that the receiver is ready to accept messages. Pattern 10, called Command Reject indicates that a message has not been accepted and a re-transmission is required. Among the reasons for rejection is detection of an error in the block check characters. Pattern 01, called Receive Not Ready, indicates a wait condition existing at the receiver. The wait applies to any messages which require buffer space. Thus supervisory messages may be sent but information bearing messages may not be sent.

The operational identifiers of a Non-sequenced frame are used to effect transmission of information that violates current message sequence counts and to configure the communications network by connecting and disconnecting devices. The NSI opcode in the operations field of a Non-sequenced control frame performs this function. If the opcode is RQI, the transmitting station is requesting that the receiving station allow it to be added to the network configuration. The station in control responds with a SIM, Set Initialization Mode, command when it is ready to allow the originating station to enter the network. NSA, Non-sequenced Acknowledge, is a special Non-sequenced frame command which allows for synchronization in the Non-sequenced mode of communications. For example, NSA is the expected response to SIM. The question of which station is in control and has the authority to admit other stations to the network is determined by system parameters. When one station is programmed to be the controlling station, it may issue SNRM, Set Normal Response Mode, commands to other stations which cause them to assume operating modes in which they may not transmit unless given permission by the controlling station. To remove a station from the network, the controlling station issues the DISC, disconnect, command which causes the addressed station to refrain from participating in the communications activity until further notice. If the station wishes to begin communications prior to further notice, it may issue a ROL, Request On Line, Non-sequenced command. If a non-controlling station receives an invalid command, it replies with command reject, CMDR. The CMDR Non-sequenced command contains an information field which describes the type of error detected.

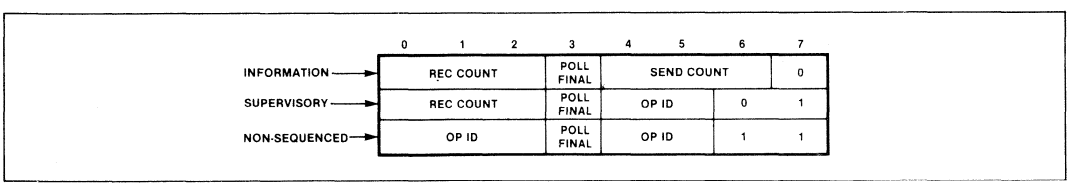


Figure 5. Control Field Format

MESSAGE INTEGRITY

Message integrity is ensured by the message format, the message rules and the data link control. The message format includes a cyclic redundancy check which can detect most transmission errors in a message. The rules of sequenced message transfer in SDLC require that the number of messages sent by one station match the number of messages received by the station to which the messages are addressed. If the message numbers are out of sequence, the station detecting the discrepancy knows to retransmit data beginning at an agreed upon message number. At the data link level, if any conditions arise requiring the re-synchronization of the framing structure, the transmitting station may abort a frame by sending at least 8 consecutive one bits followed by a flag bit pattern. The receiving station will reject the current information and reset its CRC calculation logic.

IMPLEMENTATION

The Signetics 2652 Multi-Protocol Communications Controller chip offers special support for data communications systems implemented using SDLC as the link control discipline. The chip provides recognition and insertion of FLAG patterns and recognition of the station address for secondary stations, eliminating a major preprocessing load from the host system. The chip also automatically inserts and deletes zeroes in the data stream as required by BOP protocols. The 2652 may also be employed to support character oriented protocols such as DDCMP and bisync. In loop oriented systems the controller recognizes the GA, Go Ahead, character which indicates when a station is allowed access to the link for communication with the host, and detects Abort sequences in case of message re-synchronization. The 2652 automatically generates and checks the block check characters according to CRC-CCITT. Since BOPs are independent of character length, the 2652 offers character lengths ranging from 1 to eight bits. Operating speed is up to 2 Mbits per second.

DIGITAL DATA COMM MESSAGE PROTOCOL

PROTOCOLS

Communications among data processing machines invariably become necessary when one machine produces information and another processes it. The rules governing the transfer of information from one point in a data communications network to another are called protocols. While there are many kinds of data communications protocols, there are two major categories of synchronous protocols—character oriented protocols and bit oriented protocols. The distinction between these types of communications protocols lies in the manner in which the various portions of a message are delimited.

Character oriented protocols use a partitioned alphabet (Figure 1). One part of the alphabet is used to write messages to the data communications network. The other part is used to write the data messages transferred by the network. The special part only used to write messages to the network elements is called the control set or control characters. The part of the alphabet used to write data messages is called the graphic set, or graphic characters.

One of the character oriented protocols used in data communications is the DEC Digital Data Communications Message Protocol (DDCMP) which was designed by the Digital Equipment Corporation for use in their terminal products. DDCMP is character oriented because it uses special characters which determine the nature of the communications transaction. All transactions are based upon the exchange of message sequences instead of unique characters. All special characters are embedded in the message structure. The message format separates the control messages from the graphic information. This separation provides 'transparency', that is, it allows the transmission of data patterns that have the same bit structure as the control set without violating the integrity of the protocol or the information.

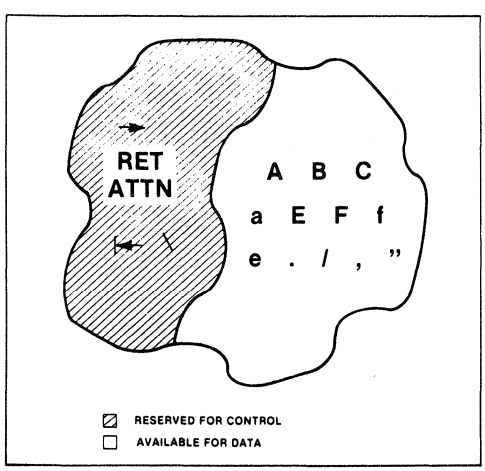


Figure 1. Partitioned Alphabet

The ability to transmit data which has the appearance of control is important to the design of a data communications network because there are cases in which the information transferred might actually look like a control character. For example, suppose that the information for which transfer is desired happened to be the patterns of ones and zeros formed by the digitization of a picture. This could easily occur if the information arose from a half-tone plot or a facsimile transmission. In this case, the pattern of light and dark points could create a series of ones and zeros that coincide with the pattern of a control character, causing an improperly designed protocol to assume unpredictable states.

To use the protocol, special send/receive devices are constructed that take appropriate actions when the outgoing message requires a communications link control character or when the incoming information contains a character from the control set.

Protocols perform the following data communications activities:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.
5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

Each of the foregoing data communications activities must be specified by one or more of the control messages in the DDCMP control set. In some cases a single message will effect the required function. In others, a function will be determined by a sequence of messages defined by successive transmissions.

OPERATING MODES

The DDCMP communications protocol features three modes of transmission. The control mode of transmission is used to control the data communications facilities. The information mode is used for transfer of information by the data communications devices using the system. The maintenance mode is used to initialize network controllers in what is called down-line loading.

Down-line loading is a technique for transferring an operating program to a device which does not feature permanent storage of its operating programs.

FORMAT

Protocol activity is often graphically represented as a plot of data transfer as a function of time. Since the data and control information usually takes the form of words, the plot is usually a sequence of horizontal bars that represents the type of information on the communication channel at a given time. In the case of synchronous communications, the blocks may be of various lengths since several words may be transmitted without a break in the transmission. If the protocol is asynchronous, the plot will consist of a series of discrete word plots, sometimes with the start and stop bits indicated on the plot.

The plots of Figure 2 illustrate the message flow resulting when using the DEC Digital Communications Message Protocol. The distinguishing feature of the plot is that the messages in one direction do not necessarily fall in the blank intervals defined by messages going in the opposite direction. This indicates that message M_i is not necessarily related to message M_j. The overlapping of messages on the time plot indicates that the protocol can be used in full-duplex mode and that the time normally required to reverse direction on a half-duplex line can be used for productive communications. However, the protocol also allows for half-duplex (alternating) operation if desired.

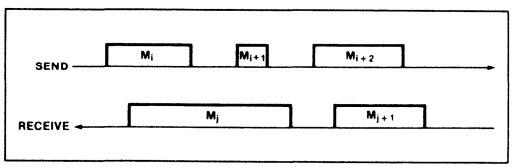


Figure 2. Message Flow

CONTROL SET

Figure 3 presents the basic DDCMP message structure. Note that the message consists of a header and an information portion. The information portion is optional but every legitimate message must contain a header. The binary code used to represent the characters is ASCII, using even parity. The SYNC bytes are eight bit characters which are used to allow the receiving circuitry to detect character boundaries.

The TYPE field contains one of the control characters to indicate what type of message is being sent. Data messages are indicated by use of the character SOH, start of header. The control messages are indicated by the character ENQ, enquiry. Maintenance messages are indicated by the character DLE, Data Link Escape. If the message is a control message, it is either an acknowledgement, ACK, or negative acknowledgement, NAK.

The Operand field consists of two sub-fields, a 14 bit MODIFIER field and a 2 bit FLAG field. If data is being transferred, the MODIFIER contains a count of the number of bytes in the information field, including the CRC bytes. This count provides transparency, since the actual bytes in the information field can be of any type. If the message is a control message, the MODIFIER clarifies the type of ACK or NAK indicated by the TYPE. There is only one type of ACK. Several types of NAK's are indicated by the MODIFIER. For example, a NAK may be due to a buffer overrun condition or a block check error on a preceding message.

The high order bit of the FLAG field flags the occurrence of a SYNC character at the end of the current message to allow the receiver to re-initialize its synchronization detection logic. The low order bit of the FLAG field indicates the current message to be the last of a series the transmitter intends to send. This allows the addressed station to begin transmission at the end of the current message.

The RESPONSE and SEQUENCE fields contain message numbers. Every station assigns a sequence number to every message it transmits. The station-assigned sequence number is placed in the SEQUENCE field. If message sequencing is lost, the control station can request the number of the last message transmitted by a certain station. When the request is received, the answering station places the last assigned sequence number in the RESPONSE field.

In a multipoint configuration, Figure 4, stations intended to receive a specified message are indicated by the entry in the ADDRESS field. The CRC 1 allows for detection of errors in the header portion of the message.

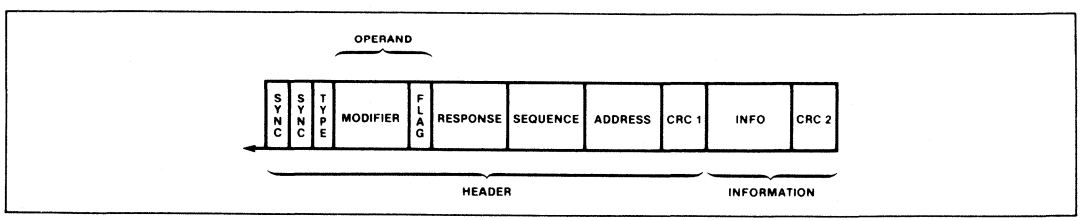


Figure 3. Message Format

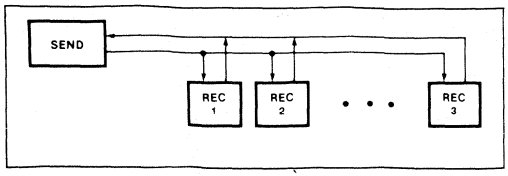


Figure 4. Multi-Point Network

The information field may contain any information the transmitting device provides, including special control characters. Proper transmission and proper system operation is assured by the DDCMP MODIFIER field which contains a count of the number of data bytes in the information field. The receiving controller loads the count and allows the appropriate number of bytes to pass before resuming active participation in the data link control. At the end of the data count, the error detection logic can be activated to check the CRC 2 field.

ERROR DETECTION AND RECOVERY

To assure integrity of the message transfer, the DDCMP messages append a two-character cyclic redundancy check (CRC) field at the end of the header field and at the end of the information field. The CRC is derived by treating the header or information field as a polynomial, dividing it by a fixed polynomial, and using the remainder as the CRC.

In operation, the transmitter appends the CRCs to the message. The receiver re-generates the CRCs and compares them to the received CRCs to determine if an error has occurred.

When an error occurs, DDCMP sends a separate negative acknowledgment (NAK) message. DDCMP does not require an acknowledgment message for all messages. Only when a transmission error occurs or if traffic in the opposite direction is light (no data message to send) is it necessary to send a special NAK or ACK message, respectively. The number in the response field of a normal header or in either the special NAK or positive acknowledgment (ACK) message specifies the sequence number of the last good message received. For example, if messages 4, 5, and 6 have been received since the last time an acknowledgment was sent and message 6 is bad, the NAK message specifies number 5 which says "message 4 and 5 are good and 6 is bad." When DDCMP operates in the full-duplex mode, the line does not have to be turned around. The NAK is simply added to the sequence of messages for the transmitter.

When a sequence error occurs in DDCMP, the receiving station does not respond to the message. The transmitting station detects from the response field of the messages it receives (or via timeout) that the receiving station is still looking for a certain message and sends it again. For example, if the next message the receiver expects to see is 5 and it receives 6, it will not change the response field of its data messages which contains a 4. This says: "I accept all messages up through message 4 and I'm still looking for message 5."

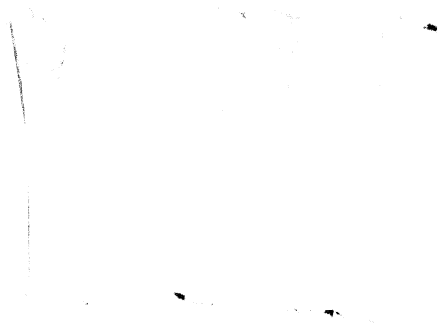
OPERATION

In operation, the transmitting station sends a Start control message to the station indicated in the Address field. The addressed station responds with a Start ACK (start acknowledge) message. The transmitting station subsequently formats and transmits successive information messages until the transaction is complete. If the receiving station is transmitting at the same time, the original transmitter will know the status of the transfer by interrogation of the Response fields in the messages directed to it. Otherwise, a special control command, Reply, may be issued to the receiving station. In this case, the receiving station will respond with a control message that indicates the last message properly received.

IMPLEMENTATION

The Signetics 2652, a multi-protocol communications controller chip, offers special support for data communications systems implemented using DDCMP as the link control discipline. The chip provides automatic SYN insertion, detection, and stripping and internal CRC generation and checking, eliminating these processing requirements from the host CPU. The 2652 operates at data rates of up to 2 Mbits per second and is also capable of supporting other synchronous character oriented protocols and bit oriented protocols such as SDLC.

When asynchronous format needs exist, the DDCMP protocol interface function can be met by use of the Signetics 2661. This Programmable Communications Interface (PCI) supports both asynchronous and synchronous line formats. It features an internal baud rate generator with 16 commonly used communications frequencies. In the synchronous mode, DDCMP can be supported with automatic SYNC generation, detection, and stripping. The CRC function is implemented with a Signetics 2653 Polynomial Generator and Checker (PGC). The 2661/2653 combination can also be utilized to be used for the IBM bisync (BSC) protocol and its derivatives.



DATA COMM ASYNC AND SYNC TRANSMISSION

ASYNCHRONOUS AND SYNCHRONOUS DATA COMMUNICATIONS

Data communications is the technology of transferring digital information from one device to another using phone lines, satellites or a combination of these and private communications lines. The main need for this type of information transfer arose when it became necessary for several computers to use the same data, for one computer to use data produced by another computer, and for a computer to transfer data between itself and its peripheral devices, such as terminals and printers. In essence, data communications is a method of passing, sharing, and disseminating information anywhere in the world via electronic technology. A simplified diagram of a data communications network is shown in Figure 1.

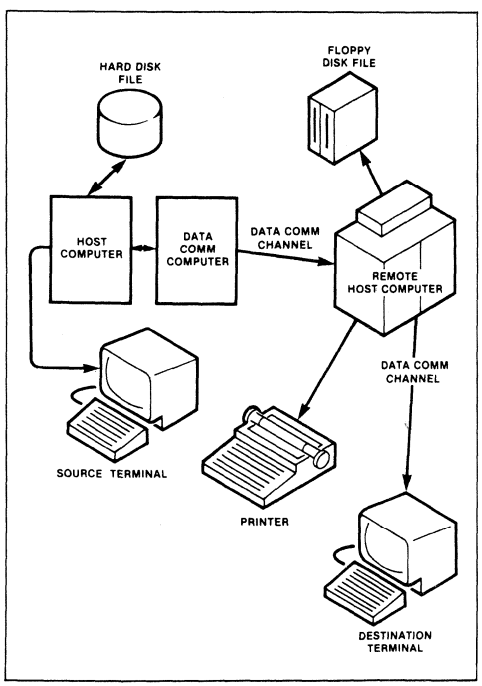


Figure 1.

DATA DEFINED

The smallest unit of data used by data processing machines is the binary digit, called the bit. A binary digit has two values, consequently two values of data can be represented by a bit. In order to deal with the large amounts of data typically communicated, groups of bits, called words, represent units of data. Thus, instead of being limited to the two values of a single bit, data processing machines can represent millions of data items by assigning a specific combination of bit values to a single data item.

For example, six bits are sufficient to represent 64 different characters. For this reason, some codes use six bits to represent the alphabet since 26 lower-case letters plus 26 capital letters ads to 52, leaving 12 combinations for the ten numerals 0 to 9. When it comes to specifying exactly where in a large manual a specific word is located, combinations of bits may be used to form addresses for the words in the document. Clearly, millions of different addresses would be required for large documents.

CLOCKS

Data processing machines usually process words rather than bits. All bits may be processed at once or they may be processed one bit at a time. If the process operates on a word, the process is called a parallel operation. If the process operates on a single bit at a time, the process is termed serial. The interval during which a bit or word is processed is called a basic (data processing) machine cycle. This interval is usually the period of a signal which continuously repeats. The repeating signal is called the clock for the data processing system.

DATA TRANSFER

Data can be transferred between parts of the same device or between two or more devices. As long as data transfers only occur from one part of a machine to another part of the same machine, the communications problems are minimal because the sending and receiving parts of the machine can be forced to be ready at the same time. Forcing such a situation is as simple as using the same clock to establish the basic processing intervals for the machines. The advantage of using the same clock to coordinate the data transfer is that the reliability of the data transfer is maximized because the data which the receiver records can be predicted to be the data that the transmitter sends, since the only variables in the transfer are predictable delays required for transmission.

Whenever the same clock is used to coordinate the transfer of data between data processing units, the transfer is called synchronous. If the data processing units use different clocks, the transfer is called asynchronous.

SERIAL TRANSMISSION

When the sending and receiving circuits are separated by a distance, from several feet to thousands of miles, it is not cost-effective to transfer data in the parallel form. Instead, serial transmission is used. Since serial operations process a single bit at a time, the cable required for a serial transfer only requires enough wires to accommodate a single bit. The usual form for such a serial channel is two wires which exhibit a difference in potential to represent the value of the bit processed.

Serial channels are more economical but less reliable than parallel channels. The reason for this loss of reliability is that cost constraints preclude addition of the cable to transfer the timing pulse that indicates when the data on the line is valid. Solutions to this problem without adding another pair of wires to the cable form the basic transmission disciplines of synchronous and asynchronous transmission techniques. If the solution includes transmission of the clock, it is called synchronous. If the solution excludes transmission of the clock, it is called asynchronous. All solutions involve a certain set of rules for how to interpret the information on the line at the receiving end of the channel. The set of rules is called a protocol.

ASYNCHRONOUS TRANSMISSION

Asynchronous transfers are widely used to allow communication between slow speed devices and a host computer. An example of such transfer occurs when keyboards and printers communicate with the host computer to which they are attached. In both instances the data is transferred at a rate which is slow in comparison to computer speeds. In addition, the number of bits transferred is relatively few so retransmission in case of errors costs little. These and other applications with similar data transfer rates are ideal for asynchronous transmission because the low data rates and the short transfers minimize the risk of mis-alignment of the time references of the sending and receiving machines.

In asynchronous transmission, the transmission medium, say two wires, is held in one state as long as data is not being transferred. When data is to be transferred, the transmitting machine signals the receiving machine that data is about to be sent by holding the line in the opposite state for a fixed length of time. After the specific interval, known to both ends of the line, the data is sent. After the message, the line is again held in its initial state for a specific time before a new message can be sent by the transmitter. The fixed intervals are measured in terms of the rate at which bits are changing at the input to the line. The initial interval is the start period and the final interval is the stop interval. These periods delimit the start and end of a word transfer. For this reason, this form of asynchronous transfer is called START-STOP transmission. The lengths of the intervals will vary from one application to another based upon the time required for the receiver to initialize itself at the beginning of a message transfer and the time required to return to its initial state at the end of a message transfer.

SYNCHRONOUS TRANSMISSION

Synchronous transmission is normally used where rate of transfer of data is high. An example of such high-speed data transfer occurs when computers must communicate or when a computer must communicate with an intelligent peripheral such as a buffered printer or word processing station. In these and similar applications, large amounts of data must be transferred in short intervals. In such applications it is inefficient to add extra start and stop intervals to each word transferred because that time could be used to transfer data bits instead.

Asynchronous techniques work because the receiver can guess reasonably accurately when each bit will arrive if it knows when the word begins. In the case of synchronous transmission data speeds, the lack of start and stop bits prevents the receiver from guessing accurately enough to allow for data to be recorded without a clock. The solution therefore is to send the clock along with the message. In some cases, the clock accompanies the data via a separate pair of wires. In other cases, the clock travels along the same pair of wires as the data through the use of special data representation schemes called self-clocking encodings.

While it is possible for synchronous techniques to dispense with the time interval required for the start and stop bits of the characters, it is not possible to dispense with the character separation function of these delimiters. Thus synchronous transmission schemes need to provide a means of indicating the specific bit at which a character begins and ends. In the case of asynchronous techniques, this was indicated by the end of the start and stop intervals. In the case of synchronous transmission schemes, the receiver is synchronized to the first character of the message using special characters. Subsequent characters are assumed to follow the initial character until the occurrence of a specified limit condition such as a count of the characters received or the receipt of another special character. The transmitted clock ensures that no characters are missed during the transfer, barring interference from outside agents.

IMPLEMENTATIONS

The Signetics 2661 and 2652 are controller chips that allow data communications systems designers to implement solutions to their communications needs in either asynchronous or synchronous formats. The 2661,

a programmable communications interface, operates in either synchronous or asynchronous modes. Since the chip interfaces directly to most eight-bit processors, the user may program, via software, the controller to handle either format. If operating in the asynchronous mode, the user may select from three stop bit options and a character length varying from five to eight bits. Assuming that the systems designer is using a standard synchronous protocol, say Bisync, the 2661 will unburden his host processor by detecting and inserting special control characters required for the protocol.

In synchronous data communications systems, the Signetics 2652 formats, transmits and receives data compatible with five major communications protocols. This monolithic communications controller handles bit oriented protocols as well as word oriented protocols, performing such functions as detection of special protocol control words such as SYNC, FLAG, GA. These characters must be interpreted at some point in data communications systems. By handling the interpretation of the general characters, the 2652 allows the host processor to be more free to handle higher level tasks. In bit-oriented protocols, the 2652 transforms the data to ensure that no data words have the appearance of message delimiters. This feature, called bit stuffing, is especially important since every character must be checked and changed if it looks like a control flag. At the receiving end, the 2652 removes the extra bits to ensure that the received data agrees with the transmitted data.

For asynchronous format requirements, the Signetics 2681 provides two independent receiver/transmitter pairs in a single package. Included are bit rate generators for popular communications speeds, fully programmable data formats, and a counter/timer circuit for auxiliary use.

SYNCHRONOUS DATA LINK CONTROL (SDLC)

PROTOCOLS

Communications among data processing machines invariably become necessary when one machine produces information and another processes it. The rules governing the transfer of information from one point in a data communications network to another are called protocols. While there are many kinds of data communications protocols, there are two major categories of synchronous protocols—character oriented protocols and bit oriented protocols. The distinction between these types of communications protocols lies in the manner in which the various portions of a message are delimited.

Character oriented protocols use a partitioned alphabet (Figure 1). One part of the alphabet is used to write messages to the data communications network. The other part is used to write the data messages transferred by the network. The special part only used to write messages to the networks elements is called the control set or control characters. The part of the alphabet used to write data messages is called the graphic set, or graphic characters.

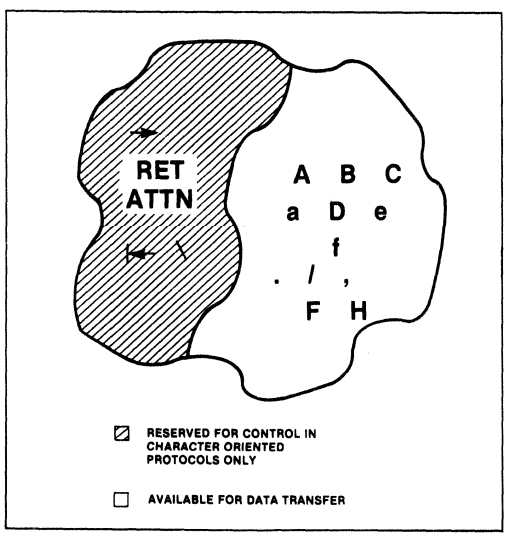


Figure 1. Transmission Alphabet

Bit-oriented protocols process information at the bit level rather than the character level to delimit message format. No characters are reserved for the control set, and only one bit pattern, 01111110, called the FLAG, is reserved. The FLAG delimits the beginning and end of a frame. Within the frame, positional significance is used to identify fields of the frame. All characters may appear at any part of the transmitted message without causing disruption of the link control hardware. In addition, since the transmission control is implemented at the bit level, there is absolutely no dependence upon the word width used. While some protocols require characters of a specified length, bit-oriented protocols are equally efficient regardless of the character length. A further advantage of bit-oriented protocols is that they cleanly separate the control of the transmission path (link) from the control of the devices using the transmission path. This latter advantage arises from the fact that control characters are used by devices while the link uses bits.

One of the most widely used bit-oriented protocols in data communications is the Synchronous Data Link Control Protocol. This protocol was designed by the International Business Machines company for use in their terminal products. The protocol serves the needs of products that must communicate in a network structure but do not feature the same character sets or lengths. In separating the link control from the device control, the protocol also makes it possible to create a layered telecommunications system which may be updated in modules as the need arises without affecting portions of the system which do not require changes in functions.

As stated above, the only special bit sequence is the FLAG, which is used to identify the beginning and end of the frame. It is possible that when data is transmitted a concatenation of characters may result in the presence of a FLAG character in the bit stream which would be incorrectly identified at the receiver as an end-of-frame FLAG. For example, an ASCII 'y' followed by 'c' would result in the serial bit stream '10011111100011' (LSBs transmitted first) which contains a FLAG.

In order to circumvent the possibility of improper operation, SDLC controllers are designed to postprocess all data prior to transmission. Whenever the controller detects a sequence in the data stream that consists of six consecutive ones, an extra zero is inserted after the first five ones to ensure that all transmitted data consists of streams of information that contains at most five consecutive ones (Figure 2). At the receiving end, the serial data is preprocessed to remove a zero which follows five consecutive ones. Thus, reception of a sequence consisting of more than five consecutive ones, subsequent to the preprocessor, signals an error condition or a control (FLAG) sequence.

Protocols perform the following data communications activities:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.

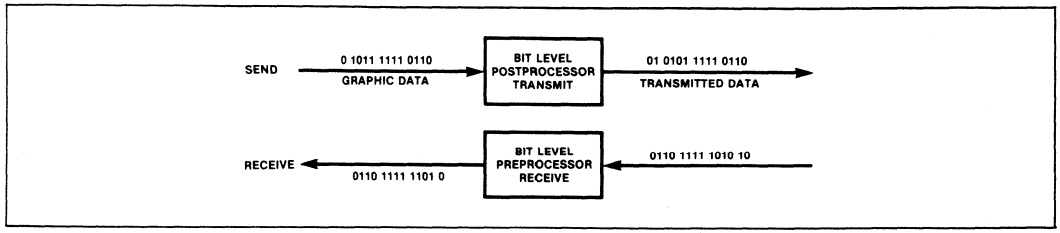


Figure 2. Bit Level Processing

5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

Each of the foregoing data communications activities is accommodated at the link level in SDLC. Path specification occurs at a higher level in the communications system. Specific characters effect communications functions only to the extent that the link must be controlled. Other communications functions are defined in the message structure of the Synchronous Data Link Control protocol.

FORMAT

Protocol activity is often graphically represented as a plot of data transfer as a function of time. Since the data and control information usually takes the form of words, the plot is usually a sequence of horizontal bars that represents the type of information on the communication channel at a given time. In the case of synchronous communications, the blocks may be of various lengths since several words may be transmitted without a break in the transmission. If the protocol is asynchronous, the plot will consist of a series of discrete word plots, sometimes with the start and stop bits indicated on the plot.

The plot in figure 3 shows the message flow occurring using the Synchronous Data Link Communications Protocol. It is significnt to note that transmitted messages and received messages may be enroute at the same time. This indicates that the SDLC protocol supports two-way communication on separate channels. This mode of communications is called full-duplex operation. However, the protocol also allows for half-duplex (alternating) operation is desired. Another feature to notice is that the incoming and outgoing messages have message numbers attached. In SDLC a receiving station may allow up to seven messages to be outstanding before acknowledgement. This mode of operation allows large blocks to be transmitted without waiting for a response from the receiver. Message M_i is not necessarily related to message M_j.

MESSAGE FORMAT

In granting access to the physical channel, SDLC features three modes of transmission. Supervisory mode is used for controlling aspects of the data link. Information transfer mode is used to effect the transfer of user data. Non-Sequenced mode is used to initialize and configure the communications system.

The SDLC message format is shown in Figure 4. Different portions of the message are determined by a character's positioning with respect to the Flag which signals the receiver that a message is ending or beginning. The header portion of the message consists of the first two bytes following the flag and the message block check character consists of the final two bytes of the message preceding the ending flag.

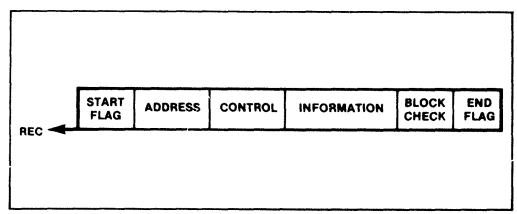


Figure 4. Message Format

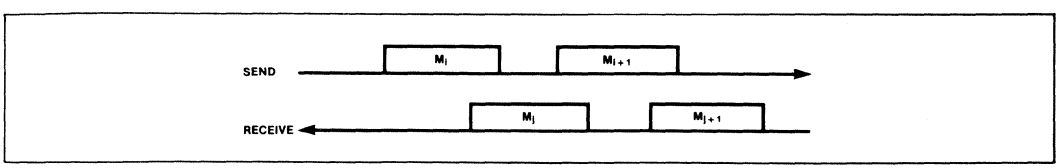


Figure 3. Message Flow

ADDRESS AND BLOCK CHECK FIELDS

The Address field identifies the station intended to respond to or receive the message.

INFORMATION FIELD

The Information field contains data to be transferred by the message. The Information field is optional. The Block Check field is a two byte cyclic redundancy check field.

CONTROL FIELD

The first three bits of the control field (Figure 5) specify a receive count for a supervisory frame or an information transfer frame. In a non-sequenced transfer, they simply modify the transaction. The fourth bit of all control fields is an indication of whether the current message is the last of a series of messages and whether the receiver is allowed to transmit data or required to respond to control messages.

The next three bits of the control field specify the number of messages received for an Information frame. For a Supervisory or Non-sequenced frame, the next two bits form an operation identifier field. The remaining bits of the control field are reserved. In the case of the Information frame, the bit following the message received count must be zero. The two bits following the two bit operation field in the Supervisory and Non-sequenced frames must be 01 to indicate a supervisory frame and 11 to indicate a Non-sequenced frame.

The operational identifiers of a supervisory frame are used to acknowledge transmissions, and request retransmissions. Pattern 00, called Receive Ready, indicates that the receiver is ready to accept messages. Pattern 10, called Command Reject indicates that a message has not been accepted and a re-transmission is required. Among the reasons for rejection is detection of an error in the block check characters. Pattern 01, called Receive Not Ready, indicates a wait condition existing at the receiver. The wait applies to any messages which require buffer space. Thus supervisory messages may be sent but information bearing messages may not be sent.

The operational identifiers of a Non-sequenced frame are used to effect transmission of information that violates current message sequence counts and to configure the communications network by connecting and disconnecting devices. The NSI opcode in the operations field of a Non-sequenced control frame performs this function. If the opcode is RQI, the transmitting station is requesting that the receiving station allow it to be added to the network configuration. The station in control responds with a SIM, Set Initialization Mode, command when it is ready to allow the originating station to enter the network. NSA, Non-sequenced Acknowledge, is a special Non-sequenced frame command which allows for synchronization in the Non-sequenced mode of communications. For example, NSA is the expected response to SIM. The question of which station is in control and has the authority to admit other stations to the network is determined by system parameters. When one station is programmed to be the controlling station, it may issue SNRM, Set Normal Response Mode, commands to other stations which cause them to assume operating modes in which they may not transmit unless given permission by the controlling station. To remove a station from the network, the controlling station issues the DISC, disconnect, command which causes the addressed station to refrain from participating in the communications activity until further notice. If the station wishes to begin communications prior to further notice, it may issue a ROL, Request On Line, Non-sequenced command. If a non-controlling station receives an invalid command, it replies with command reject, CMDR. The CMDR Non-sequenced command contains an information field which describes the type of error detected.

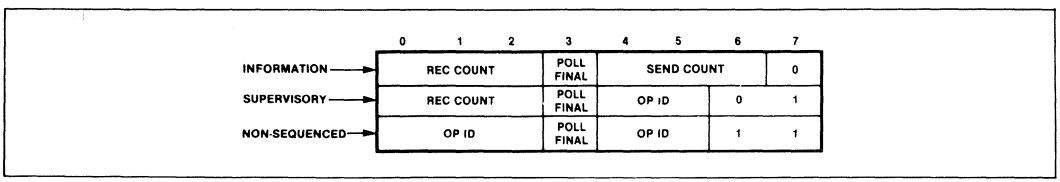


Figure 5. Control Field Format

MESSAGE INTEGRITY

Message integrity is ensured by the message format, the message rules and the data link control. The message format includes a cyclic redundancy check which can detect most transmission errors in a message. The rules of sequenced message transfer in SDLC require that the number of messages sent by one station match the number of messages received by the station to which the messages are addressed. If the message numbers are out of sequence, the station detecting the discrepancy knows to retransmit data beginning at an agreed upon message number. At the data link level, if any conditions arise requiring the resynchronization of the framing structure, the transmitting station may abort a frame by sending at least 8 consecutive one bits followed by a flag bit pattern. The receiving station will reject the current information and reset its CRC calculation logic.

IMPLEMENTATION

The Signetics 2652 Multi-Protocol Communications Controller chip offers special support for data communications systems implemented using SDLC as the link control discipline. The chip provides recognition and insertion of FLAG patterns and recognition of the station address for secondary stations, eliminating a major preprocessing load from the host system. The chip also automatically inserts and deletes zeroes in the data stream as required by BOP protocols. The 2652 may also be employed to support character oriented protocols such as DDCMP and bisync. In loop oriented systems the controller recognizes the GA, Go Ahead, character which indicates when a station is allowed access to the link for communication with the host, and detects Abort sequences in case of message resynchronization. The 2652 automatically generates and checks the block check characters according to CRC-CCITT. Since BOPs are independent of character length, the 2652 offers character lengths ranging from 1 to eight bits. Operating speed is up to 2 Mbits per second.

BINARY SYNCHRONOUS COMMUNICATIONS (BSC)

PROTOCOLS

Communications among data processing machines invariably become necessary when one machine produces information and another processes it. The rules governing the transfer of information from one point in a data communications network to another are called protocols. While there are many kinds of data communications protocols, there are two major categories of synchronous protocols—character oriented protocols and bit oriented protocols. The distinction between these types of communications protocols lies in the manner in which the various portions of a message are delimited.

Character oriented protocols use a partitioned alphabet (Figure 1). One part of the alphabet is used to write messages to the data communications network. The other part is used to write the data messages transferred by the network. The special part only used to write messages to the network elements is called the control set or control characters. The part of the alphabet used to write data messages is called the graphic set, or graphic characters.

One of the most widely used character oriented protocols in data communications is the Binary Synchronous Communications Protocol (Bisync or BSC). This protocol was designed by the International Business Machines company for use in their terminal products in the early 1960's.

To use the protocol, special send/receive devices were constructed that took the appropriate actions when the outgoing message required the insertion of a communications link control character or when the incoming information contained a character from the control set that indicated special action.

As long as the information portion of the message only used characters from the graphic set, Bisync worked well. However, there were cases in which the information that was transferred might violate the groundrules and include graphic information which actually looked like a control character. For example, suppose that the information for which transfer is desired happened to be the patterns of one and zeros formed by the digitization of a picture. This could easily occur if the information arose from a half-tone plot or a facsimile transmission. In this case, the pattern of light and dark points could create a series of ones and zeros that coincide with the pattern of the control character.

In order to circumvent the possibility of improper operation in the controllers designed to interpret the control characters, the graphic set included a special character, called a data link escape (DLE) character. Whenever the controller detected this character followed by an 'STX' control character in the incoming data stream, it would ignore control characters in the incoming data until the next data link escape character. Reception of a single data link escape character followed by a legitimate character from the control set indicates the transmission of a control character by the sending controller. Reception of a control character not preceded by a single data link escape character merely indicates the transmission of information. This sub-mode of the BSC protocol is called the transparent mode.

Data Link protocols perform the following data communications activities:

1) Establish a connection between the sender and receiver.
2) Grant the sender access to the physical channel.
3) Define the message formats and data blocking rules.
4) Provide message acknowledge and recovery procedures.
5) Provide rules for time-sharing of the channel (called line turnaround procedures).
6) Define the methods of coordination between sender and receiver.
7) Provide for supervisory and transfer activities.

Each of the foregoing data communications activities must be specified by one or more of the characters in the BSC control set. In some cases a single character will effect the function. In others, a function will be defined by a sequence of transmissions.

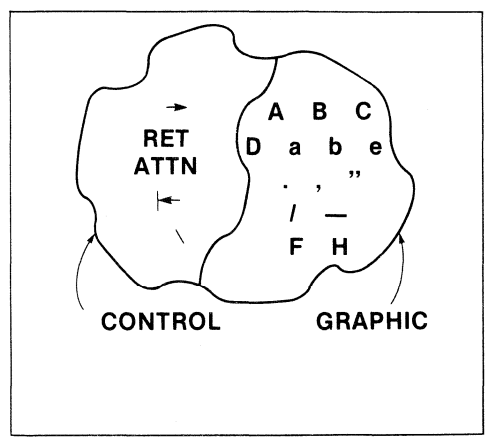


Figure 1. Partitioned Alphabet

OPERATING MODES

To grant the sender access to the physical channel, the Binary Synchronous Communications protocol features two modes of transmission. The non-transparent mode of transmission is the mode of transmission in which all information transferred only uses characters from the graphic set. The transparent mode is the case in which the graphic information is allowed to contain characters from the control set.

FORMAT

Protocol activity is often graphically represented as a plot of data transfer as a function of time. Since the data and control information usually takes the form of words, the plot is usually a sequence of horizontal bars that represents the type of information on the communication channel at a given time. In the case of synchronous communications, the blocks may be of various lengths since several words may be transmitted without a break in the transmission. If the protocol is asynchronous, th plot will consist of a series of discrete word plots, sometimes with the start and stop bits indicated on the plot.

The plot below (Figure 2) depicts message sequencing using the Binary Synchronous Communications Protocol. It is apparent that no messages are transmitted at the same time messages are being received. This characteristic of Bisync is called half-duplex operation. In this mode of operation, only one station can use the communications channel at a time. For this reason, the plot shows the received messages falling in the time intervals for which there are no transmit messages.

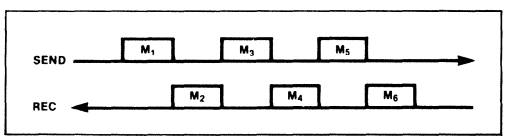


Figure 2. Message Sequencing

CONTROL SET

Every message transferred according to the BSC protocol is controlled using one or more of the special characters reserved for controlling the transfer of messages. Bisync uses fifteen control characters to effect orderly transfer of information. The binary code used to represent the codes varies according to the data format used, i.e., ASCII, EBCDIC, etc. Regardless of the code used, unique characters are reserved for the following functions:

ACK0	Acknowledgment of even numbered transactions
ACK1	Acknowledgment of odd numbered messages
ARQ	Automatic Request for retransmission
DLE	Data Link Escape
ENQ	Enquiry
EOT	End of Transmission
ETB	End of Transmission Block
ETX	End of Text
ITB	Intermediate Text Block
NAK	Negative Acknowledge
RVI	Reverse Interrupt
SOH	Start of Header
STX	Start of Text
SYN	Synchronize
TTD	Temporary Text Delay
WACK	Wait for Acknowledgment

The BSC message format is shown in Figure 3. Different portions of the message are delimited by control characters which signal the receiver of the next action to expect or perform. The header portion of the message contains user defined information. Some of this information may be address and status data. In Bisync, the header is an optional feature. It is not necessary that every correct implementation of Bisync include a header.

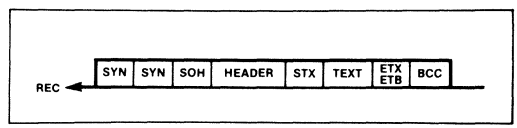


Figure 3. Message Format

A message session begins with the sending station issuing a request for a session by transmission of an ENQ character (Figure 4). If the receiving station is not ready to receive, it responds with a NAK character. If the sending station continually requests a communications session, when the reciever is ready to accept, it responds with an ACK character. The transmitting station subsequently begins with the transfer of messages in the format shown in Figure 3. The session ends with the transfer of an EOT character.

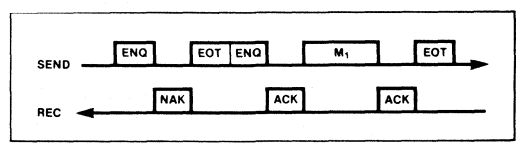


Figure 4. Sample Message

When the message contains information that may be mistaken as control characters by the receiving controller, BSC offers the transparent mode of transmission. The transparent mode of data transfer is the mode in which the receiver ignores all characters until notified

to resume observing the control characters. This mode of transfer begins when the transmitter prefixes a DLE character to the STX character at the beginning of the text portion of the message (Figure 5). When the receiver detects the DLE-STX, it ignores the remaining control characters of the message until a message delimiter occurs which is preceded by a DLE character.

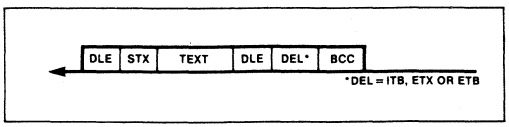


Figure 5. Transparent Format

Since the DLE character is the means of entering and exiting the transparent mode of communication, special care must be taken to ensure that DLE characters occurring in the data do not cause premature exit from the transparent mode. The technique for accomplishing this is to pre-process all the data and insert DLE characters in front of each of the data characters having the same format as the DLE character. The receiver strips the inserted DLE and stores the original DLE.

At the end of the information transfer, the ending delimiter is inserted as usual with the exception that the delimiter is preceded by a DLE character. When the receiver detects the delimiter preceded by the DLE, it resumes control character interpretation, treating the next two characters as the block check character (BCC).

MESSAGE INTEGRITY

To assure integrity of the data transfer, all Bisync text messages append a two-character block check character (BCC) field at the end of the message. The BCC is derived by treating the message as a polynomial, dividing it by a fixed polynomial, and using the remainder as the BCC.

In operation, the transmitter appends the BCC to the message. The receiver regenerates the BCC and compares it to the received BCC. If they match, an ACK0 or ACK1 is sent back and the transmitting station proceeds with the next message. If the BCCs don't match, the receiver returns a NAK and the transmitter re-sends the last message. If more than a predetermined number of NAKs are received, the transmitting station assumes a faulty condition and alerts the operating system for appropriate action.

The first SOH or STX to follow a line turnaround resets the BCC logic. All succeeding STXs or SOHs until the next line turnaround are included in the block checking. ITB, ETB, and ETX are always included in the BCC and are followed by the block check characters associated with the portion of the message in which they occur. In the transparent mode of transmission, false ITB, ETB and ETX characters are ignored since they are not preceded by DLE. The actual message terminator characters are preceded by DLE in the transparent modes. The actual terminators are included in the BCC calculations in the transparent mode but the extra DLEs inserted to ensure transparent transmission are excluded. SYN characters used as line fill are not included in the BCC.

IMPLEMENTATION

The Signetics 2661, a universal asynchronous/synchronous data communications controller chip, offers special support for data communications systems implemented using BSC as the link control discipline. The chip provides automatic SYN or DLE-SYN insertion and stripping, eliminating a major preprocessing load from the host system. Both transparent and normal modes of transmission are accommodated by the chip. Since Bisync may be implemented with several character sets, the Signetics 2661 operates with equal facility on character lengths ranging from five bits to eight bits.

The Signetics 2653 Polynomial Generator/Checker is a polynomial generator and checker which provides character level checking and message checking. The Binary Synchronous Communications protocol may be used with parity checking on each character or a group of characters, or it may be used with the CRC-12 or CRC-16 polynomials for cyclic redundancy checking. The Signetics 2653 implements both types of parity checking and both cyclic redundancy check polynomials. This chip is fully compatible to the normal and transparent modes of Binary Synchronous Communications protocol operation, and automatically excludes or includes characters in the accumulation as required by the protocol. Additionally, the 2653 can detect the receipt of control characters and signify the controlling microprocessor of their occurrence.

DATA COMMUNICATIONS ERROR CONTROL (DCEC)

PROTOCOLS

Whenever information is transferred from one physical location to another, it is often the case that data arriving at the intended destination differs from the data sent. This is an unavoidable consequence of information transfer. Since some of the data arrives without modification some of the time, the problem is one of probabilities. Error control is often a matter of adding clues to messages to allow the receiver to answer the question: Is the data at the receiver the same as the data that left the transmitter?

If the receiver has perfect information concerning the transfer, the entire message can be reliably reconstructed at the receiver. In such a case, the penalties of errors during transmission can be avoided. These cases are covered in the discipline of error correction. In the absence of perfect knowledge, the clues in the transmitted message might be sufficient to determine that the data received is not the same as the data transmitted. This is embodied in the error detection discipline. Error detection is the foundation on which error correction is based.

In data communications, error control is practiced at the physical level and at the message level. The physical level includes the interface between the data communications network and the devices using the network for transfer of data. The physical level also includes the physical representation of the data as it is being transferred through the communications network. The message level includes the various protocols used to transfer blocks of data from one point in the network to another.

The amount of error control required for a data communications network is often a function of the amount of error correcting capability of the receiver. For example, humans often detect errors based upon the context of the information transferred. If the word "context" had been spelled "cintext", a human reader would have correctly guessed that the data intended should have been "context". In some applications, it is possible to take advantage of the fact that a human is in the loop. This mode of error control is sometimes acceptable in low-speed terminal communications. The method of control is to effect a rule (protocol) in which the receiving machine transmits back to the terminal every character which the terminal sends to the host. When the character arrives back at the terminal, the human verifies that no error occurred during transmission by comparing what appears on the screen or paper to what was entered (Figure 1). Although there are some flaws in this method, they are easily overcome by retransmission and the cost of the system is minimal.

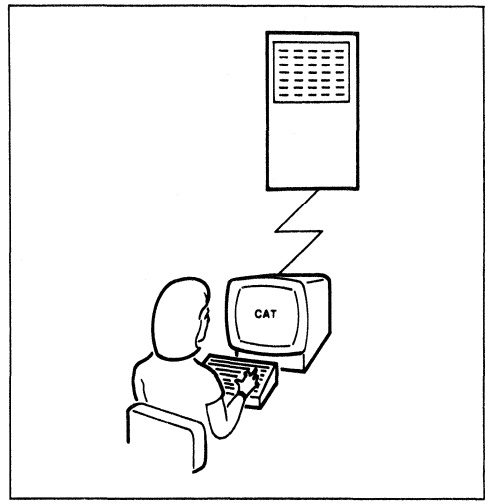


Figure 1. Human Error Detection

SYSTEMS INTEGRITY

In cases where ignoring data communications errors does not give acceptable data communications performance, the wise approach is to include the least amount of control consistent with effective communications. The reason for the minimization guideline is that error control improves in proportion to the amount of capital invested and the amount of electronics dedicated to the task. Since all error control is probabilistic (i.e., you can't catch every error), increasing levels of protection come at a price that is disproportional to the added coverage.

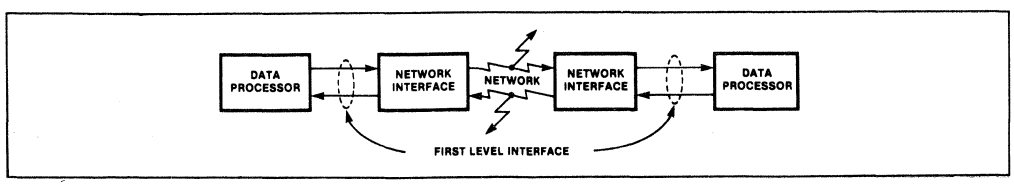


Figure 2. Interface to the Data Communications Network Should be Secure

Violation of this law of diminishing returns is often observed at the interface between the data communications network and the systems it serves. The first line of error control exists at the interface between the data communications network and the using devices. Yet, it is often the case that reliable data transmission is requested within a data communications system but no care is taken to ensure that the data offered to the network is free of error (Figure 2). If the data transferred features a parity error at the input to the system, the receiving device will receive incorrect data, faithfully transferred by the communications network, in the case of an error-free transmission.

It is often a simple matter to add a parity check to data within the source/sink devices to ensure data integrity on a systems basis. In this manner, needless error recovery procedures will not be activated to isolate network errors that do not exist.

The next level of error control is to design data transfer protocols which guarantee that the messages transferred by the system contain enough redundancy to ensure that errors are detected and recovery procedures begun (Figure 3). In many cases, the cause of the error is of a temporary nature. Consequently, retransmission is sufficient to recover since the disturbing phenomenon will have ceased. When multiple messages must be transmitted, each message should have a system assigned number to ensure that errors can be detected early and that retransmission can be limited to the erroneous portions. Modern protocols, such as SDLC and DDCMP, feature message numbers to allow for the detection of loss of messages and the loss of the sequences of messages. BiSync, the original synchronous protocol developed by IBM in the 1960's features an optional header field in which such information might be incorporated.

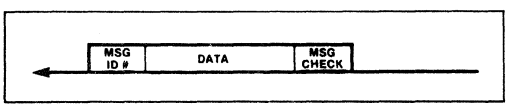


Figure 3. Reliability Incorporated in Message Structure

ERROR CHECKING AND RECOVERY

An important function of a line protocol is to assure correct reception of data. Communications facilities are error-prone. To compensate for this, line control procedures include the generation, transmission, and testing of check bits. These check bits (often called Block Check Characters—BCC) make up the trailer field *of the transmission block.* They are generated by a checking algorithm which is usually applied only to the information field of a block.

Each block of data transmitted is error-checked at the receiving station in one of several ways, depending on the code and the functions employed. These checking methods are Vertical Redundancy Checking (VRC), which is parity checking by character as the data is received; and either Longitudinal Redundancy Checking (LRC) or Cyclic Redundancy Checking (CRC), which check the block after it is received. *After each transmission, the receiving station normally replies with a positive (ACK) acknowledgment (data accepted, continue sending), or with a negative (NAK) acknowledgment (data not accepted; e.g., a transmission error was detected—retransmit previous block). The acknowledgment may be in a special control message (ACK) or in the response field in the header of the next message to be send in the opposite direction.* Retransmission of a block of data following an initial NAK is usually attempted a number of times. Until the block is accepted, the data buffer cannot be released by the transmitting system.

VRC (Vertical Redundancy Checking) is an odd or even parity check performed on a per-character basis and requires a parity check bit position in each character. If individual characters are represented by eight bits, such as when using an eight-level code, seven may be used to represent actual numbers and letters, and the eighth may be reserved for checking purposes. The presence or absence of the eighth bit provides the inherent checking feature. For example, in an even parity check, the parity bit is used to make the total number of one bits in the character even. If the character contains four zeros and three ones, then a one bit is inserted as the parity bit.

Longitudinal Redundancy Checking (LRC) is a technique for checking an entire message or block of data. In this case, an exclusive "OR" logic is used for all the bits in the message and the resulting character, called the *Block Check Character (BCC),* is transmitted as the last character in the block. The receiving device independently performs the same counting procedure and generates a Block Check Character. It then compares its own BCC character with the one received. If they are not identical, an error condition exists, and the sending device is notified that an error condition exists within the block. *LRC is frequently used in conjunction with VRC to increase the error detection capability within a system.*

Cyclic Redundancy Checking (CRC) is a more sophisticated method of block checking than LRC. *This type of error checking involves a polynomial division of the data stream by a CRC polynomial. The 1's and 0's of the data become the coefficients of the dividend polynomial while the CRC polynomial is preset. The division uses subtraction modulo 2 (no carries) and the remainder serves as the Cycle Redundancy Check.* The receiving station compares the transmitted remainder with its own computed remainder, and an equal condition indicates that no error has occurred.

There are many constants that may be used to perform the CRC division. Two of the most popular versions are called *CRC-16* (which uses a polynomial of the form $x^{16}+x^{15}+x^2+1$) and *CRC-CCITT* (which uses a polynomial of the form $x^{16}+x^{12}+x^5+1$). Each generates a 16-bit BCC. CCITT, the International Consultative Com-

mittee for Telephony and Telegraphy, is responsible for usage standards.

Error checking also involves checking for sequence errors. Protocols handle this in different ways. These include alternating acknowledgments and block sequencing. The technique used depends on the protocol. The receiving station sends back an indication of a sequence error with a negative acknowledgment or some other control message.

IMPLEMENTATION

Signetics offers a comprehensive set of error control features in its data communications controller chips, the 2652, the 2653 and the 2661. This chip set is completely compatible, allowing the designers of data communications systems the ability to implement error control at the system level and at the protocol level. The Signetics 2653 Polynomial Generator/Checker supports block check coded error detection and parity generation and checking, featuring CRC-16 and CRC-12 which are able to detect 99 percent of the burst errors occurring up to the length of the code used. Since the 2653 operates in the parallel mode, it can accomplish parity generation and checking on the data bus of the system using the data communications network, thereby providing the first line of protection for systems level data integrity, as indicated in Figure 2.

Within the data communications network, the entire family of Data Communications Controllers support the latest protocols and feature special functions which allow for the accurate transmission of messages formatted according to both bit-oriented protocols and character-oriented protocols. Violations of the message structure are detected by special functions built into the chips to account for automatic detection and checking of the block check characters in BiSync messages (Signetics 2661). Message delimiting and detection is automatically accomplished by the programmable features of the Signetics 2652 which provides automatic bit-stuffing and deletion to ensure that transmitted messages adhere to bit level protocol. Special control patterns, Flag, Go Ahead, and Abort, are generated and detected by the Signetics 2652 as it automatically monitors the link level control characters of SDLC and similar advanced protocols.

The Signetics 2652 and the Signetics 2661 both feature a maintenance mode of operation which allows for isolation of errors on a node basis. This mode of system verification is extremely important for cases in which it is necessary to determine whether the receiver is at fault, the transmitter is at fault or the channel is at fault. This method of system verification is called Loop-back. In operation, the output of the transmitter is directed to the receive circuitry (Figure 4) so that the transmitting station can verify that no errors appear in the data it is sending.

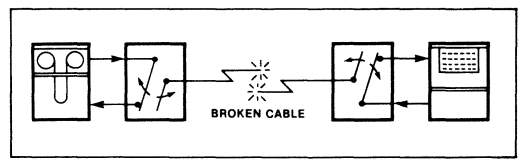


Figure 4. Detecting a Channel Fault in Maintenance Mode

FEATURE

A DESIGNER'S REVIEW OF DATA COMMUNICATIONS

Efficient communication systems depend on knowledge of design concepts and principles for encoding and transmitting digital data

Alex Goldberger Signetics Corporation
811 E Arques Ave, Sunnyvale, CA 94086

Recent developments in information systems and computer and microcomputer hardware have highlighted the need for efficient data communications. Industrialists, educators, financial institutions, and government organizations are finding computer services essential to their operation, and the data communications link is an integral part of these services.

Data communication refers to the electronic transmission of encoded information or data from one point to another. As used here, the term encompasses all the physical elements, systems, devices, and procedures that are required for the transmission and reception of data between two or more points. Elements of a data communication system are communication channels, transmission modes, line conditioning, modems, serial communication interfaces, data link configurations, information codes, and protocols.

The data communication process generally requires at least five elements: a transmitter or source of information, a message, a binary serial interface, a communication channel or link, and a receiver of transmitted information (Fig 1). A data communications interface is often needed to make the binary serial data compatible with the communication channel.

Communication Channels and Facilities

A communications link or channel is a path for electrical transmission between two or more stations or terminals. It may be a single wire, a group of wires, a coaxial cable, or a special part of the radio frequency spectrum. The purpose of a channel is to carry information from one location to another. All channels have limitations on their information handling abilities, depending upon their electrical and physical characteristics.

There are three basic types of channels: simplex, half-duplex, and full-duplex. As an example of each, consider transmission between points A and B in Fig 1.

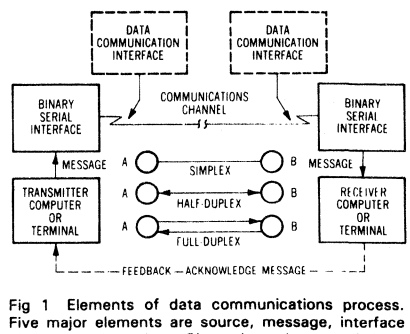


Fig 1 Elements of data communications process. Five major elements are source, message, interface channel, and receiver. Channel can be any of three basic types

Reprinted with permission
from Computer Design—May, 1981 issue

Transmission from A to B only (and not from B to A) requires a *simplex* channel. Simplex channels are used in loop mode configurations such as supermarket checkout terminals. Transmission from A to B and then from B to A, but not simultaneously, requires a *half-duplex* channel. If a 2-wire circuit is used, the line must be turned around to reverse the direction of transmission. A 4-wire circuit eliminates line turnaround. Transmission from A to B and from B to A simultaneously describes a *full-duplex* channel. Although four wires are most often used, a 2-wire circuit can support full-duplex communications if the frequency spectrum is subdivided into receive and transmit channels.

In addition to the direction of transmission, a channel is characterized by its bandwidth. In general, the greater the bandwidth of the assigned channel, the higher the possible transmission speed. This speed is usually measured in terms of the number of line signal elements per second, the *baud rate*. If a signal element represents one of two binary states, the baud rate is equal to the bit rate. When more than two states are represented, as in multilevel modulation, the bit rate exceeds the baud rate. The range of channels includes private wire, wideband, Digital Data Service, limited distance, voice grade, subvoice grade, and telegraph (Table 1).

Digital vs Analog Transmission

Digital transmission can be applied to digital data or analog voice signals. In either case, information is sent over the communications channel as a stream of pulses. Pulses transmitted over a communication line are distorted by line capacitance, inductance, and leakage. The longer the line or the faster the pulse rate, the more difficult it is to interpret the received signal. This signal degradation is the reason for the closely spaced regenerative repeaters used in digital data transmission facilities. When noise and distortion threaten to destroy the integrity of the pulse stream, the pulses are detected and regenerated. If the regeneration process is repeated properly, the received signal will be an exact replica of the transmitted signal. It is possible to transmit pulses over short distances using privately owned cable or common carrier wire pairs. This is *baseband transmission* and usually requires line drivers and receivers on each end of the line. Longer distance communication must use the digital transmission facilities of the common carriers.

In analog transmission, a continuous range of signal amplitudes or frequencies is sent over the communications line. Linear amplifiers maintain signal quality. The voice telephone network supplied by the common carriers uses analog transmission facilities to service most data communications users. To interface the analog voice channels to digital terminals and computers, a modulator-demodulator (modem) is used. In a modem, digital information modulates a carrier signal, which passes through the telephone network just as does a voice signal. At the receiving end, the signal is demodulated back into digital form.

Voice Grade Lines

Voice grade telephone lines are available through the public switched network (Direct Distance Dialed or DDD); as private leased lines without conditioning; and as private, conditioned, leased lines. Although the bandwidth is the same for all three, the effective data rates vary because of different specifications for signal noise, amplitude attenuation, and envelope delay distortion.

Dial-up lines are the 2-wire pairs supplied by the common carriers on the public switched telephone network. Most often these lines are used for half-duplex operation, although frequency band splitting modems can facilitate full-duplex at 1200 bits/s. A major advantage of dial-up lines is that any point on a worldwide telephone network can be reached. Furthermore, communication costs are limited to the time the lines are actually in use.

Four major problems are associated with the switched telephone network. First, the lines may be noisy. The human brain can interpret what is being said despite the interference that plagues many calls, but computers and terminals can easily lose or misinterpret data because of noise. Second, delay distortion is caused by the various frequency components of a signal being transmitted at a nonuniform speed through the transmission medium. This may result in received data that are erroneous. Third, the switched network requires relatively long connect, disconnect, and turnaround times, which limit the system data throughput. Fourth, the reliability of telephone switching equipment is relatively low.

Although more costly than dial-up lines, private leased lines largely circumvent the problems that afflict the switched network. Their basic advantages are ready availability and freedom from busy signals, fixed monthly charges, and conditioning for better data quality, as well as higher transmission rates and throughput. Leased lines are generally 4-wire circuits usable for half- or full-duplex operation. Simultaneous transmission and reception is possible, and line turnaround is eliminated. The basic disadvantages of leased lines are higher cost and the line's connection to only one location. However, if telecommunication demands entail high volume, high quality, high speed traffic between two points, a leased line is the best choice.

Digital Data Services

The Bell System has developed a digital transmission network that provides higher data rates with fewer errors at a lower cost than conventional analog transmission facilities. Known as Dataphone Digital Service (DDS), the network is available in 32 U.S. cities and recently has been granted Federal Communications Commission (FCC) approval for 64 other cities. Two point, full-duplex, private line service is provided at synchronous data rates of up to 1.544M bits/s.

In June 1977, the FCC approved construction of American Telephone & Telegraph (AT&T)'s Dataphone Switched Digital Service (DSDS) for 27 cities at data rates

TABLE 1

Communication Channel Characteristics

Channel Type	Channel Interface	Data Rates (bits/s)	Applications
Fiber optics	Fiber optic connector	Up to 10M	Computer-computer, Computer-high speed peripheral
Private wire or cable	Line drivers and receivers Modem eliminators Limited distance modems	1M to 2M	In-plant data communications
Wideband analog	Wideband Bell 300-series modems, CCITT V-series wideband modems	19.2k to 230.4k	Telephone channel multiplexing
Dataphone Digital Service	Data Service Unit Digital Station Terminal	2.4k, 4.8k, 9.6k, 56k, 1.544M	Private terminal-computer geographically dispersed links
Switched telephone network (DDD)	2-wire modems	0 to 2k (async), 2k to 4.8k (sync)	Terminals, data collection stations, other interactive communications
	Acoustic couplers	300, 450, 600, 1.2k (async)	
Leased voice grade lines (with or without conditioning)	2/4 wire modems	0 to 2.4k (async) 2k to 9.6k (sync)	Remote batch, private communications networks
Subvoice grade	Narrowband modems	150 to 200	Teletypes, A-D converters, telemetry
Telegraph	dc signaling	45 to 75	TWX, TELEX

of 56k bits/s. Operations will not begin until AT&T files tariffs for DSDS and proposes an accounting system. Both of these measures must then be approved by the FCC. AT&T has also proposed that DDS and DSDS be included in its heralded Advanced Communications Service message switching network.

Specialized common carriers offer a variety of services in addition to those of the Bell System. These include shared private line services such as EXECUNET by MCI Communications and SPRINT by Southern Pacific; satellite services by the Radio Corporation of America, Western Union, and others; packet-switching carriers including Telenet by General Telephone and Electronics and Tymnet by Tymshare; and facsimile and electronic mail services such as Graphnet, TWX, TELENET, and SPEEDFAX.

Modems

Modems are devices that convert digital data from a computer or terminal to a modulated carrier waveform required by the communication channel. One modem is needed at each end of the channel, as shown in Fig 2. Modems are also known as data sets and are designed for specific kinds of service and for specific bandwidths or data rates. Those discussed here accept a binary serial input from the transmitter and provide a binary serial output to the receiver. Parallel input modems (used mostly for paper tape transmission) and analog input machines (used primarily for facsimile transmission) are not considered. The three types that are considered are short haul, wideband, and voice grade (Table 2).

Short haul modems operate over relatively short distances—generally less than 10 mi (16 km)—on solid conductor, non-band limited, non-loaded lines. In some cases, they are not true modems but are line drivers and line receivers that transmit and receive digital data. Although the communication line must be carefully chosen, the cost can often be one-tenth that of a voice grade modem rated at the speed. Other advantages are higher speed and reliability and easier maintenance.

Wideband modems operate over telephone transmission facilities at speeds of 19.2k bits/s to 230.4k bits/s. This class of data set is supplied almost exclusively by the common carrier and requires the bandwidth of 6 to 60 dedicated voice grade channels. Examples are the Bell 303B, -C, and -D for 19.2k-, 50k- and 230.4k-bit/s full-duplex operation.

Voice grade telephone lines with a bandwidth of 2700 Hz (300 to 3000 Hz) are by far the most common medium used for data communications. A voice grade modem, designed for use on these lines, should be selected on the basis of the type of service (dial-up or leased), the required data rate, and an acceptable level of error performance. The two broad categories of voice grade modems are asynchronous units and synchronous units. Asynchronous units operate at a maximum data rate of 1800 bits/s over dial-up facilities and 2000 bits/s

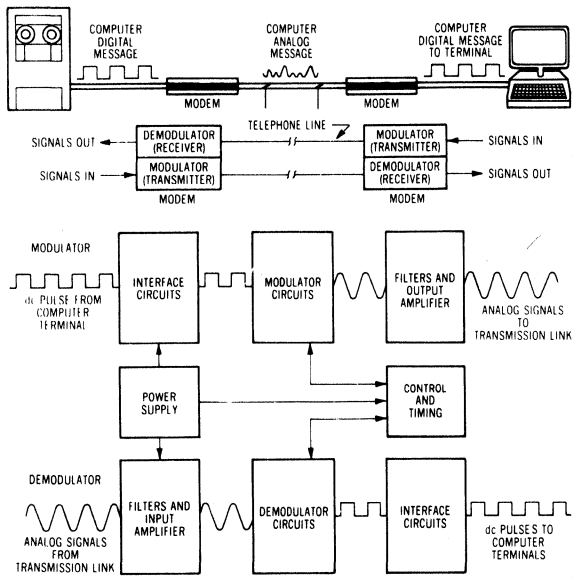


Fig 2 Analog transmission using modems. Modem is divided into two functional segments: modulator and demodulator. Modulator accepts digital input and converts squared dc pulse voltages to analog, audio signals, which are sent over transmission link. At end of link, demodulator of second modem reconverts analog signals to digital output

on conditioned leased lines. Acoustic couplers are asynchronous modems designed for dial-up use that are generally limited to speeds of 600 bits/s or less. Synchronous units operate at a maximum data rate of 4800 bits/s over dial-up and 9600 bits/s on conditioned leased lines.

Asynchronous and Synchronous Transmission

Asynchronous data are typically produced by low speed terminals with rates of less than 1200 bits/s. In asynchronous systems [Fig 3(a)], the transmission line is in a mark (binary 1) condition in its idle state. As each character is transmitted, it is preceded by a start bit, or transition from mark to space (binary 0), which indicates to the receiving terminal that a character is being transmitted. The receiving device detects the start bit and the data bits that make up the character. At the end of the character transmission, the line is returned to a mark condition by one or more stop bits, and is ready for the beginning of the next character. (An asynchronous character varies in length depending on the information code employed.) This process is repeated character by character until the entire message has been sent. The start and stop bits permit the receiving terminal to synchronize itself to the transmitter on a character by character basis.

Synchronous transmission [Fig (3b)] uses an internal clocking source within the modem to synchronize the transmitter and receiver. Once a synchronization character (SYN) has been sensed by the receiving terminal, data transmission proceeds character by character without the intervening start and stop bits. The incoming stream of data bits is interpreted on the basis of the receive clock supplied by the modem. This clock is usually derived from the received data through a phase locked loop. The receiving device accepts data from the modem until it detects a special ending character or a character terminal count at which time it knows that the message is over. The message block usually consists of one or two synchronization characters, a number of data and control characters (typically 100 to 10,000), a terminating character, and one or two error control characters. Between messages, the communication line may idle in SYN characters or be held to mark. Note that synchronous modems can be used to transmit asynchronous data, and, conversely, asynchronous modems can be used for synchronous data if the receiving terminal can derive the clock from the data.

Asynchronous transmission is advantageous when transmission is irregular (eg, when it is initiated by a keyboard operator's typing speed). It is also inexpensive because of the simple interface logic and circuitry required. Synchronous transmission, on the other hand, makes far better use of the transmission facility by eliminating the start and stop bits on each character. Furthermore, synchronous data are suitable for

TABLE 2

Modem Characteristics

Modem Type	Communications Channel	Data Rate (bits/s)	Use
1. Voice grade (vg)			
a. High speed synchronous	Leased line (3002) Dial-up	4.8k, 7.2k, 9.6k 4.8k	High volume machine to machine communications. 600 to 1200 bits/s
b. Medium speed synchronous Medium speed asynchronous Medium speed asynchronous	Leased line/dial-up Leased line Dial-up	2.4k, 3.6k 1.8k, 2k 1.2k	Interactive or low speed remote batch operations. 150 to 300 bits/s
c. Low speed asynchronous	Dial-up	300, 600	Interactive teleprinters and glass teletypewriters, data acquisition and collection. 30 to 60 bits/s
2. Wideband			
a. Super group (60 vg) b. Group (12 vg) c. Half group (6 vg) d. Lineplexer (2 vg)	5700, 5800 (TELPAK) 8801 8803 2 leased lines	230.4k 40.8k, 50k, 56k 19.2k 19.2k	Large volume telephone line multiplexing, dedicated computer to computer links
3. Short haul			
a. Limited distance [<10 mi (<16 km)]	Private wire/cable Nonloaded, non-conditioned, non-carrier line	2k to 1M 2k to 19.2k	Data communications in plant (private wire) or off premises where distance is <10 mi (<16 km) [(leased line)]
b. Medium distance [<50 mi (<80 km)]	Leased line	2k to 9.6k	Intermediate distance [10 to 50 mi (16 to 80 km)]
4. Modem eliminators and line drivers/receivers	Private wire/cable	2k to 1.544M	On-premises data communications. Typical distances are 500 ft (152 m) to 2 mi (3.2 km)

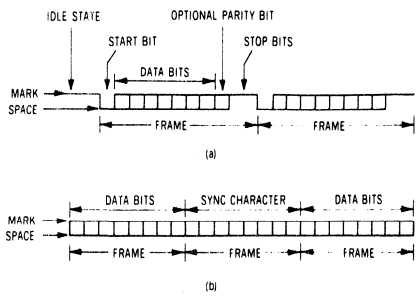


Fig 3 Asynchronous and synchronous transmission. Choice of proper timing mode depends directly on application. Asynchronous transmission (a) is used mostly with man-machine interfaces; synchronous transmission (b) offers high speed necessary for machine-machine communication

multilevel modulation, which combines two or four bits in one signal element (baud). This can facilitate data rates of 4.8k- or 9.6k bits/s over a bandwidth of 2.4 kHz. Synchronous modems offer higher transmission speeds, but are more expensive because they require precisely synchronized clock and data.

Modulation Techniques

Whether to use the dial-up network or leased lines depends on how the modem modulates data prior to sending them over the phone line. Certain modulation techniques permit higher transmission rates than others, and all modulation techniques directly affect the maximum data rate and the error performance. The three basic modulation techniques are frequency shift keying (FSK), amplitude modulation (AM), and phase modulation (PM) (Fig 4).

The most popular form of frequency modulation is FSK, in which the carrier frequency (operating at, say, 1700 Hz) is modulated ±500 Hz to present binary 1 or binary 0. Thus, a frequency of 1200 Hz represents a

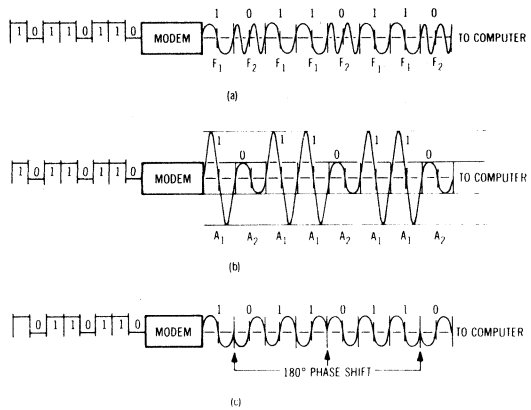


Fig 4 Modulation techniques. In frequency shift keying (a), digital signal (left) received by modem is shifted in frequency to represent binary 1 and binary 0. Because of the time needed to shift from one frequency to another, two is generally the maximum number used. In amplitude modulation (b), digital signals are amplitude modulated by modems using this technique. Here, one carrier signal amplitude (A_1) represents binary 1, and another (A_2), binary 0. In phase modulation (c), whenever 180° phase shift is encountered, receiving mode assigns value of binary 0; at all other times, value of binary 1 is assumed

zero, while a frequency of 2200 Hz represents a binary 1. FSK techniques are generally quite suitable for low speed devices like teleprinters and allow operation at speeds as high as 1800 bits/s.

AM enables a modem to transmit and receive the analog equivalents of binary 1s and 0s. This technique involves varying the amplitude of the line's carrier frequency. Several levels of amplitude modulation are possible, allowing twice as much data to be sent in the same time frame. Both AM and FSK are quite suitable for data transmission; however, FSK has a noise advantage over AM, and AM allows more efficient use of the available bandwidth.

PM modems are generally described in terms of the number of phase shifts generated, and operate at speeds of 2000 bits/s and above. In this technique, the transmitted signal is shifted a certain number of degrees in response to the pattern of bits coming from the terminal or computer. For example, in a 2-phase PM modem, if the analog signal generated by the transmitting modem is shifted 180°, a binary 1 (or 0 if desired) is indicated. If there is no shift, then the signal will be interpreted as a series of zeros (or ones) until such a shift is sensed. Generally, PM modems operate in four and eight phases, permitting up to two or three times the data to be sent over the line in the same bandwidth. Most 4800- and 9600-bits/s modems use PM.

Conditioning and Equalization

As data in the form of analog signals are sent down the line between modems, they suffer from the effects of envelope delay and amplitude distortion. Signals of different frequencies are delayed or attenuated by varying amounts as they are transmitted. To compensate for these effects, two techniques are employed: line conditioning and modem equalization.

Conditioning is the process by which the telephone company maintains the quality of a specific, privately leased line to a certain standard of permissible delay distortion and signal attenuation. AT&T has two types of conditioning referred to as C and D. There are five categories of C conditioning (C1 through C5) and two categories of D conditioning (D1 and D2). C conditioning atempts to equalize the drop in signal voltage and envelope delay for all frequencies transmitted; D conditioning controls the signal to noise ratio and harmonic distortion. Both may be used on the same communication channel.

Equalization refers to modem compensation for amplitude and envelope delay distortion of the line. Equalization is seldom required in lower-speed modems attached to a leased line, since minimum line conditioning is sufficient. However, conditioning and equalization are required when higher speed modems (4.8k- and 9.6k bits/s) are attached. Modems used for high speed transmission over the dial-up network must have equalization, since it is never certain exactly which unconditioned telephone line will be used.

Communication Line Sharing and Modem Sharing

When several input/output devices are required at one end of a communication link, a multiplexer or modem sharing unit, which enables these devices to share one communication line, can be used to reduce costs. Multiplexers take low speed inputs from a number of terminals and combine them into one high speed data stream for simultaneous transmission on a single channel. At the other end of the link, a second multiplexer (actually a demultiplexer) reconverts the high speed data stream into a series of low speed inputs to the host computer. The channel is split into time slots (time division

multiplexing) or frequency bands (frequency division multiplexing). Intelligent or statistical multiplexers increase line utilization by allocating time slots on the basis of a line activity algorithm.

Modem sharing units enable multiple terminals to share one modem. They are particularly valuable in networks that require clusters of terminals at remote sites because the number of modems and transmission lines are reduced. Operation is polled half-duplex. Multiport modems can split a high speed channel (eg, 9600 bits/s) into various medium speed channels (eg, two 2400 bits/s and one 4800 bits/s), thus permitting several medium speed terminals to share a 9600-bit/s line. A miniplexer is a device that performs channel splitting for DDS as well as for a single 3002 leased line. A lineplexer or biplexer splits 19.2k-bit/s data into two 9600-bit/s paths that can be transmitted over two conditioned full-duplex channels. This eliminates the need for a wideband channel to send and receive data at 19.2k bits/s. A port sharing unit connects to a communication controller or central processing unit port and transmits or receives data from two to six terminals or modems. Less costly than a multiplexer, it reduces the number of controller ports in a polled terminal data communications configuration and makes more efficient use of connected ports.

Standards and Protection

The electrical, functional, and physical interface to data terminal equipment provided by modems is compatible with Electronics Industries Association (EIA) or International Consultative Committee for Telephone and Telegraph (CCITT) standards. Most commercial models conform to EIA RS-232, and plug to plug compatibility via a 25-pin connector is ensured between modems and data terminal equipment that subscribe to this standard. CCITT V.26 is the electrical equivalent of RS-232-C, while V.24 is the U.S. standard's functional pin equivalent. CCITT V.35, a current-mode, 34-pin connector interface standard for serial data transmission up to 56k bits/s, is used by wideband European modems and in the Bell System DDS Data Service Unit at 56k bits/s. Military standard (MIL-STD) 188 is a U.S. government standard for military communications equipment. An improved EIA functional standard, RS-449, was approved in November 1977. Although not yet implemented in U.S. modems, it is being incorporated into modems used in Germany and in the CCITT V.36 modem.

Common carrier equipment on the switched telephone network must be protected. A device called a data access arrangement (DAA) limits the attached modem's signaling power to prevent it from exceeding the power level restrictions of the communication channel. In 1977, the FCC ruled that modem manufacturers can incorporate equivalent protective circuitry in their products, register them with the FCC, and connect them directly to the telephone network. DAAs are available from FCC-certified independent suppliers and can be leased from the Bell System. Modems rented from the Bell System or those used on leased lines do not require a DAA.

Protocols

Protocols provide the necessary ground rules to ensure the orderly and accurate transfer of data between digital devices. Data communications protocols are growing in importance as the terminal population increases, distributed processing becomes widespread, and new communications technologies, such as packet switching and satellite links, become commonplace.

Protocols associated with data communications have several major levels, or layers, that define various functions and operations. Each level is designed to be functionally independent of the others, but the function of each depends on the correct operation of the previous level. The protocols embodied in these levels range from those that define the physical and electrical links, eg, RS-232-C and CCITT V.35, to those that are responsible for functions such as message buffering, code conversion, recognition and reporting of faulty conditions in terminals or lines, communication with the host mainframe, and management of the communication network. They are implemented by software packages like International Business Machines (IBM)'s Systems Network Architecture (SNA), CCITT's X.25, and Digital Equipment Corporation (DEC)'s DECnet.

The remainder of this article concerns data link control protocols (DLCs), the sets of rules necessary for effective communication between terminals and computers over conventional communications channels. DLCs are involved in handling the communications link itself and moving information across it efficiently and accurately. Their basic functions are to establish and terminate a connection between two stations; to ensure message integrity through error detection, requests for retransmission, and positive or negative acknowledgments; to identify sender and receiver through polling or selection; and to handle special control functions such as requests for status, station reset, reset acknowledge, start, start acknowledge, and disconnect.

Structure of Data Link Controls

Data link controls can be classified into byte control protocols (BCPs) and bit oriented protocols (BOPs). In BCPs, a defined set of communication control characters effects the orderly operation of the data link. These control characters are part of a character code set. BCP messages are transmitted in blocks composed of a header or control field, a body or text field, and a trailer or error checking field with characters used as field or block delimiters. Examples of BCPs are IBM's Binary Synchronous Communications Protocol (BISYNC) and DEC's Digital Data Communications Message Protocol (DDCMP). Block formats for these are illustrated in Fig 5.

BOPs use only two or three specific control characters for operation of the data link. These characters are used to delimit the beginning (FLAG) and end (FLAG, ABORT, GA) of a message frame. Upon receipt of the opening FLAG, positional significance is used to delineate the bit sequence that follows into prescribed fields (Fig 5).

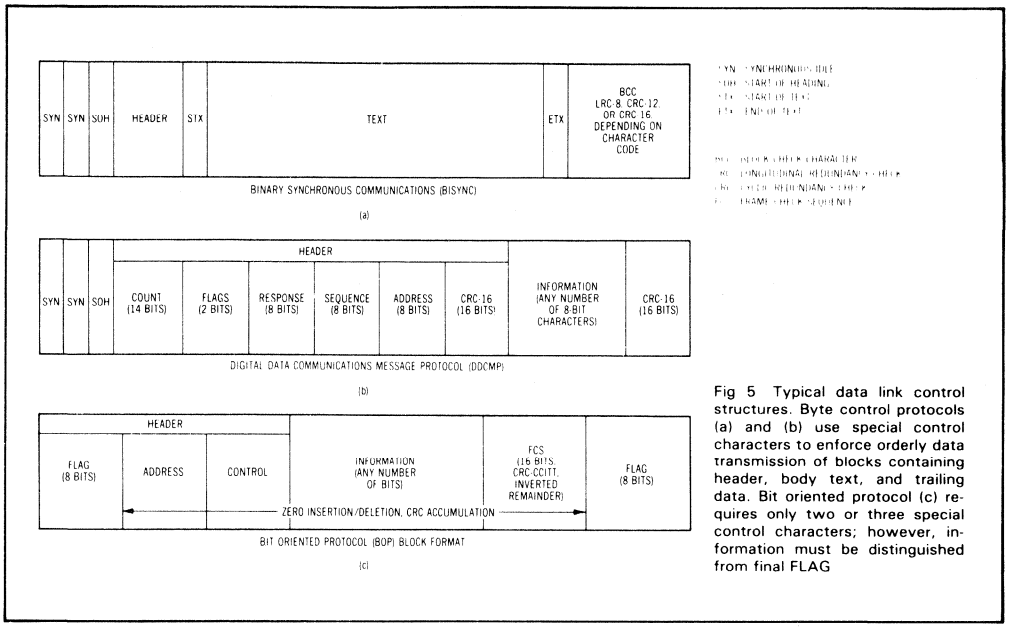


Fig 5 Typical data link control structures. Byte control protocols (a) and (b) use special control characters to enforce orderly data transmission of blocks containing header, body text, and trailing data. Bit oriented protocol (c) requires only two or three special control characters; however, information must be distinguished from final FLAG

These fields are address, control, information, and frame check sequence. The address, control, and frame check fields are of fixed length; the information field length is variable and may be zero.

BCP Messages

As already stated, BCP messages are transmitted in units called blocks. The header field contains auxiliary information that identifies the address of the message destination or source, the job number (if any), the type of message (data or control), the control action, and a positive or negative acknowledgment to ensure error-free reception of a previous message (or messages). Control actions are used to reset or initialize a secondary station, to acknowledge good or bad reception of blocks, to inquire why a response or acknowledgment has not occurred within a specific time period, or to abort a transfer sequence. The control information is conveyed via special characters or character sequences.

The text field contains any data being transmitted. The text may be characters of the information code set or may be transparent to that code set. In the latter case, pure data (binary, packed decimal, floating point), specialized codes, or machine language computer programs must be distinguished from characters in the code set being used. This is done by employing a transparent mode whose implementation depends on the specific DLC.

To ensure correct reception of information over communication facilities, a sequence of check bits, often called block check characters or BCCs, are generated and transmitted as an error check field. Each block of data transmitted is checked for errors at the receiving station in one of several ways, depending on the code and functions employed. These checking methods include vertical redundancy check (VRC), a parity check on each character, in conjunction with a longitudinal redundancy check (LRC), a horizontal parity; and cyclic redundancy check (CRC), which involves a polynomial division of the bit stream by a CRC polynomial.

BOP Messages

BOPs are more straightforward and universal than the BCPs just discussed. BOP messages are also transmitted in frames, and all messages adhere to one standard frame format. Common characteristics of BOPs are the independence of codes, line configurations, and peripherals; the use of positional significance instead of control characters or character counts; one standard frame format for all messages; the possibility of half- or full-duplex operation; the achievement of information transparency through zero insertion and deletion; and error checking on a complete frame.

A frame starts with the 8-bit FLAG sequence, followed in order by the sequences ADDRESS, CONTROL, INFORMATION (if present), and FRAME CHECK, and ends with another FLAG sequence. Each station attached to the data link continuously searches for the FLAG sequence and an ADDRESS sequence. In multipoint operation, for example, a secondary station must detect a FLAG immediately followed by its own ADDRESS to enable the receiver.

TABLE 3

Common Protocol Characteristics

Feature	BISYNC	DDCMP	SDLC	ADCCP
Full duplex	No	Yes	Yes	Yes
Half duplex	Yes	Yes	Yes	Yes
Message format	Variable	Fixed	Fixed	Fixed
Link control	Control character, character sequences, optional header	Header (fixed)	Control field (8 bits)	Control field (8/16 bits)
Station addressing	Header	Header	Address field (8 bits)	Address field (8 bits to 00)
Error checking	Information field only	Header, information field	Entire frame	Entire frame
Error detection	VRC/LRC-8 VRC/CRC-16	CRC-16	CRC-CCITT	CRC-CCITT
Request for retransmission	Stop and wait	Go back N	Go back N	Go back N, selected reject
Maximum frames outstanding	1	255	7	127
Framing—start —end	2 SYNs Terminating characters	2 SYNs Count	Flag Flag	Flag Flag
Gaps between characters allowed	Yes	No	No	No
Information transparency	Transparent mode	Inherent (count)	Inherent (zero insertion/ deletion)	Inherent (zero insertion/ deletion)
Control characters	Numerous	SOH, DLE, ENQ	None	None
Character codes	ASCII EBCDIC Transcode	ASCII (control character only)	Any	Any

When the primary station transmits, the station ADDRESS sequence, which is usually one 8-bit field, designates which secondary station is to receive the balance of the transmitted frame. When a secondary station transmits, the ADDRESS tells the primary station which secondary station originated the frame. A secondary station must recognize its valid address before it can accept a frame and take any action on the contents of that frame. Also, the primary station will accept a frame only when it contains the address of a secondary station that has been given permission to transmit. To ensure the integrity of the data being transmitted, the ADDRESS sequence appears within each frame. This enhances flexibility in that the primary station can interleave receptions from several secondary stations without intermixing individual station information transfers.

The CONTROL field follows the ADDRESS sequence and is composed of one or two 8-bit bytes, depending on the protocol implementation. It is the heart of the BOP message, for it determines the type of message, the send and receive frame sequence counts, and a poll command from the primary station or final response from the secondary station. The primary station uses CONTROL to tell (command) the addressed secondary station what operation to perform. The secondary station uses CONTROL to react (respond) to the primary station.

The INFORMATION field may vary in length; this includes different lengths in the sequential frames making up a complete transmission. The data may be configured in any code structure: straight binary, binary coded decimal, and packed decimal, among others. However, the content of the field must be self-defining by actual or implied means. For example, peripheral device control characters, such as carriage return, will actually be part of the INFORMATION field, while the code being used may be implied in the address of a specific terminal designed for a specific code. Furthermore, whether a frame contains an INFORMATION field at all depends on the particular CONTROL format transmitted. table 3 presents a comparison of common DLCs.

Synchronization Techniques

Four kinds of synchronization—bit, character, block and message—must be distinguished when using synchronous transmission. Bit synchronization is achieved through a received clock signal which is coincident with the received serial data stream. Most modems or "business machines" (ie, terminals) derive this clock by means of phased lock loops from the 0 to 1 and 1 to 0 transitions occurring in the received data. This technique, called self-clocking, overcomes the effect of propagation delay between distant stations and the tendency of electronic circuits within the modem to drift.

Character synchronization is accomplished by recognizing one or two "phasing" characters, often called SYN or sync characters. The receiver senses these SYN characters and phases its receive logic to recognize, by bit count, the beginning and end of each subsequent character. To ensure character synchronization throughout a message, SYN sequences are sometimes inserted in the transmitted data stream at 1- or 2-second intervals. This permits receiving stations to verify that they are in sync.

Request for Retransmission

As previously mentioned, DLCs include an error checking field to allow the receiving station to validate the message. When errors are detected, the receiving station issues a request for retransmission (ARQ). The two types of ARQs are *stop and wait* and *continuous*. Each provides defined methods for acknowledging correct (error free) reception of transmitted blocks of information.

When a connection is established in the stop and wait ARQ, the transmitter sends one block and then stops. Eventually, the receiver acquires that block, subjects the block to an error check, and then sends an ACK control character to the transmitter to indicate that the block is correct, or a NAK control character to indicate an error. If an ACK is returned, the transmitter sends the next block in sequence. If a NAK is returned, that block is retransmitted. Thus, the stop and wait mode involves periods of idleness, including propagation delays between each block, so that the line is not communicating nearly at its rated capacity.

In continuous ARQ, the transmitter keeps sending one block after another without stopping. The receiver and transmitter retain individual counts of the blocks outstanding and provide buffer storage to retain those blocks. Only when an erroneous block is detected does the receiver tell the transmitter to resend that block and all subsequent in-transit, but unacknowledged, blocks.

Summary

As the installed base of computers and the speed and volume of their output have increased, so has the need to transmit that output to more places over longer distances. Inherent in the data communications process—the electronic transmission of encoded information from one point to another—are various physical elements, devices, and systems, as well as standards and procedures. An understanding of these basic elements and concepts can help users of computer services to take advantage of the communication systems that are now available.

Bibliography

H. C. Folts and H. R. Carp, *Compilation of Data Communications Standards,* McGraw-Hill, New York, NY, 1978

L. C. Hobbs, *Computer and Data Processor Technology Newsletter,* April 1980

J. C. McQuillan and V. G. Cerf, *A Practical View of Data Communications Protocols,* IEEE Press, New York, NY, 1978

Signetics Data Communications Handbook, Signetics Corp, Sunnyvale, Calif, 1980

A. J. Weissberger, "Simplify Serial Data Links with LSI Chips," *EDN Magazine,* October 20, 1978; "Data-Link Control Chips: Bringing Order to Data Protocols," *Electronics Magazine,* June 8, 1978; "Data Communications," Parts 1 to 5, *Electronic Design Magazine*, 1979

About the Author:

As applications manager for the microprocessor division of Signetics Corporation, Alex Goldberger is responsible for product applications support, new product planning, and technical marketing support for microprocessors, microcomputers, and associated peripheral circuits. He has worked with integrated circuits since 1969 and was the design manager for the universal asynchronous receiver/transmitter (UART), the first large scale integration circuit for data communications. He holds a BSEE from the City College of New York and an MSEE from the Polytechnic Institute of Brooklyn, and is the author of several articles for technical publications.

VIDEO DISPLAY

FOR DETAILED INFORMATION SEE RELEVANT DATA BOOK OR DATA SHEET

SINGLE-CHIP COLOUR CRT CONTROLLER (EUROM)

GENERAL DESCRIPTION

The SAA5350 EUROM is a single-chip VLSI NMOS crt controller capable of handling all display functions required by the CEPT videotex terminal, model A4. Only minimal hardware is required to produce a videotex terminal using EUROM – the simplest configuration needs just a microcontroller and 4 Kbytes of display memory.

Features

- Minimal additional hardware required
- Screen formats of 40/80 character by 1-to-25 row display
- 512 alphanumeric or graphical characters on-chip or extendable off-chip
- Serial attribute storage (STACK) and parallel attribute storage
- Dynamically redefinable character (DRCS) capability over full field
- Interfaces with 8/16-bit microprocessors with optional direct memory access
- On-chip scroll map minimizes data to be transferred when scrolling
- On-chip colour map RAM (4096 locations) and three on-chip digital-to-analogue converters allow 32 colours on-screen
- On-chip digital-to-analogue converters are non-linear to compensate for crt non-linearity
- Memory interface capable of supporting multi-page terminals. EUROM can access up to 128 Kbytes of display memory
- Programmable cursor
- Programmable local status row
- Three synchronization modes:

stand-alone	built-in oscillator operating with an external 6 MHz crystal
simple slave	directly synchronized from the source of text composite sync
phase-locked slave	indirect synchronization allows picture-in-text displays (e.g. VCR/VLP video with text overlay)

- On-chip timing composite sync output
- Zoom feature which allows the height of any group of rows to be increased to enhance legibility

PACKAGE OUTLINE

40-lead DIL; plastic (SOT-129).

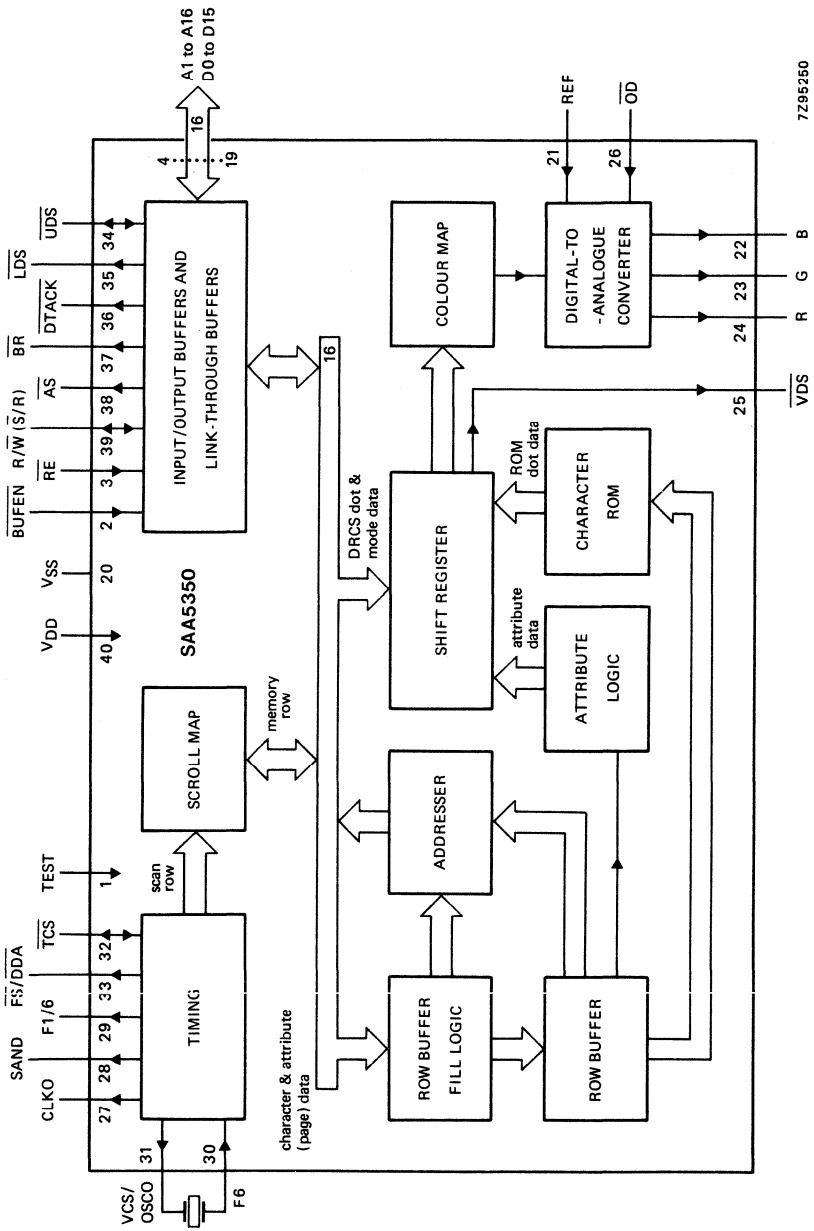


Fig. 1 Block diagram.

Video Attributes Controller (VAC)

Originally published by Signetics February 1985

Product Specification

DESCRIPTION

The Signetics 2673A and 2673B Video Attributes Controllers (VAC) are bipolar LSI devices designed for CRT terminals and display systems that employ raster scan techniques. Each contains a high speed video shift register, field and character attributes logic, attribute latch, cursor format logic and half dot shift control.

The VAC provides control of visual attributes on a field or character by character. Internal logic preserves field attribute data from character row to character row so that an attribute byte is not required at the beginning of each row. The 2673B provides for reverse video, blank (non-display), blink, underline and highlight attributes and a graphics mode attribute to work in conjunction with the Signetics 2670 Display Character and Graphics Generator (DCGG). The 2673A substitutes a light pen (strike-thru) attribute for the graphics attribute.

The horizontal dot frequency is the basic timing input to the VAC. Internally, this clock is divided down to provide a character clock output for system synchronization. Up to ten bits of video dot data are parallel loaded into the video shift register on each character boundary. The video data is shifted out on three outputs at the dot frequency. On the VIDEO output, the data is presented as a three level signal representing low, medium and high intensities. The three intensities are also encoded on two TTL compatible video outputs. Light or dark screen background can be selected.

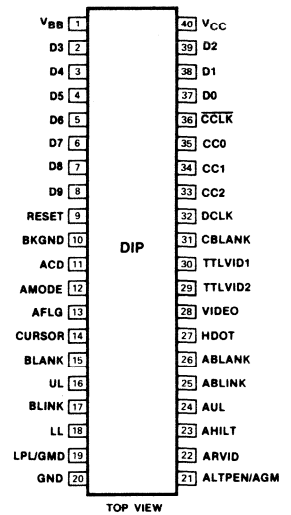
FEATURES

- **18 and 25MHz video dot rate**
- **Three level current driven video output**
- **Three level encoded TTL video outputs**
- **Character/field attribute logic:**
  - **Reverse video**
  - **Character blank**
  - **Character blink**
  - **Underline**
  - **Highlight**
  - **Light pen strike-thru or graphics control**
- **Field attributes extend from row to row**
- **Light or dark field**
- **Cursor reverse video logic**
- **Up to 10 dots per character**
- **Composite blanking for light field retrace**
- **Optional field graphics control output**
- **High speed bipolar design**
- **TTL compatible**
- **Compatible with Signetics 2672 PVTC and 2670 DCGG**

APPLICATIONS

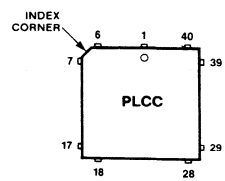
- **CRT terminals**
- **Word processing systems**
- **Small business computers**

PIN CONFIGURATION



TOP VIEW

CD00490S



TOP VIEW

CD0044PS

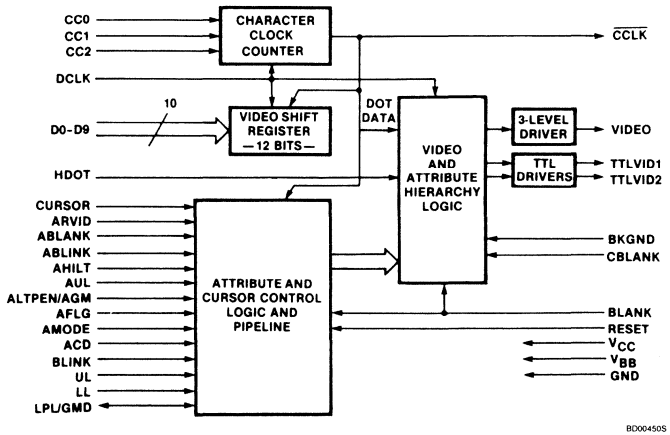
Pin	Function	Pin	Function
1	NC	23	NC
2	V_{BB}	24	ALTPEN/AGM
3	D3	25	ARVID
4	D4	26	AHILT
5	D5	27	AUL
6	D6	28	ABLINK
7	D7	29	ABLANK
8	D8	30	HDOT
9	D9	31	VIDEO
10	RESET	32	TTLVID2
11	BKGND	33	TTLVID1
12	NC	34	NC
13	ACD	35	CBLANK
14	AMODE	36	DCLK
15	AFLG	37	CC2
16	CURSOR	38	CC1
17	BLANK	39	CC0
18	UL	40	$\overline{CCLK}$
19	BLINK	41	D0
20	LL	42	D1
21	LPL/GMD	43	D2
22	GND	44	V_{CC}

Product Specification

ORDERING CODE

PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C			
	Graphics Attribute		Light Pen Attribute	
	25MHz	18MHz	25MHz	18MHz
Ceramic DIP	SCB2673BC5I40	SCB2673BC8I40	SCB2673AC5I40	SCB2673AC8I40
Plastic DIP	SCB2673BC5N40	SCB2673BC8N40	SCB2673AC5N40	SCB2673AC8N40
Plastic LCC	SCB2673BC5A44	SCB2673BC8A44	SCB2673AC5A44	SCB2673AC8A44

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
DCLK	32	36	I	**Dot Clock:** Dot frequency input. Video shift output rate.
$\overline{CCLK}$	36	40	O	**Character Clock:** A submultiple of DCLK. The frequency ranges from one sixth to one twelfth of DCLK, as determined by the state of the CC0 – CC2 inputs.
CC2 – CC0	33 – 35	37 – 39	I	**Character Clock Control:** The logic state on these three static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate. Character clock rates of 6 through 12 dots per character may be specified.
D0 – D9	37 – 39, 2 – 8	41 – 43, 3 – 9	I	**Dot Data Input:** These are parallel inputs corresponding to the character graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the falling edge of each character clock.
HDOT	27	30	I	**Half Dot Shift:** When this input is high, the serial video output is delayed by one half dot time. This input is latched on the falling edge of each character clock.
CURSOR	14	16	I	**Cursor Timing:** This input provides the timing for the cursor video. When high, effectively reverses the intensities of the video and attributes. Cursor position, shape, and blink rate are controlled by this input.
BKGND	10	11	I	**Background Intensity:** Specifies light or dark video during BLANK and character fields. Affects the intensities of all attributes.
BLANK	15	17	I	**Screen Blank:** When high, this input forces the video ouputs to the level specified by the BKGND input (either high or low intensity). Not effective when CBLANK is high.

Product Specification

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
CBLANK	31	35	I	**Composite Blank:** Used with the TTL video outputs only. When high, this input forces the video outputs to a low intensity state for retrace blanking. When BKGND input is low, or when using video outputs, this input may be tied low.
ARVID	22	25	I	**Reverse Video Attribute:** The intensity of the associated character or field video is reversed. All other attributes are effectively reversed.
AHILT	23	26	I	**Highlight Attribute:** All dot video (including underline) of the associated character or field is highlighted with respect to the BKGND input and the reverse video attribute.
ABLANK	26	29	I	**Blank Attribute:** Generates a blank space in the associated character or field. The blank space intensity is determined by the BKGND input, the reverse video attribute, and the CURSOR input.
ABLINK	25	28	I	**Blink Attribute:** The associated character or field video is driven to the intensity determined by BKGND and the reverse video attribute when the BLINK input is high.
AUL	24	27	I	**Underline Attribute:** Specifies a line to be displayed on the character or field. The line is specified by the UL input. All other attributes apply to the underline video.
ALTPEN/ AGM	21	24	I	**Light Pen Attribute (2673A):** Specifies a highlighted line to be displayed on the character or field. The line is specified by the LPL input.
			I	**Attribute Graphics Mode (2673B):** This input is latched and synchronized to provide a field GMD output for the 2670 DCGG.
AMODE	12	14	I	**Attribute Mode:** Specifies character (AMODE = 0) or field (AMODE = 1) attributes mode.
AFLG	13	15	I	**Attributes Flag:** The VAC samples and latches the attributes inputs when this input is high. If field attributes are specified (AMODE = 1), the attributes are double buffered on a row basis. Thus, each scan line of every character row will start with the attributes that were valid at the end of the previous row.
ACD	11	13	I	**Attributes Control Display:** In field attributes mode (AMODE = 1), if ACD = 0, the first character in each new attribute field (the attribute control character) will be suppressed and only the attributes will be displayed. If ACD = 1, the first character and the attributes are displayed. This input has no effect in character mode (AMODE = 0).
BLINK	17	19	I	**Blink:** This input is sampled on the falling edge of BLANK to provide the blink rate for the character blink attribute. It should be a submultiple of the frame rate.
UL	16	18	I	**Underline:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK.
LPL/GMD	19	21	I	**Light Pen Line (2673A):** Indicates the scan line(s) for the light pen strike-thru attribute. Latched on the falling edge of BLANK.
			O	**Graphics Mode (2673B):** This output provides a synchronized, latched, field graphics mode corresponding to the AGM input. This output can be used to control the GM input on the 2670 DCGG.
LL	18	20	I	**Last Line:** Indicates the last scan line of each character row. Used internally to extend field attributes across row boundaries. Latched on the falling edge of BLANK. This input has no effect in character mode (AMODE = 0).
VIDEO	28	31	O	**Video:** A three level serial video output which corresponds to the composite dot pattern of characters, attributes and cursor.
TTLVID1	30	33	O	**TTL Video 1:** This output corresponds to the serial, non-highlighted video dot pattern.
TTLVID2	29	32	O	**TTL Video 2:** This output corresponds to the highlighted serial video dot pattern. Should be used with TTLVID1 to decode a composite video of three intensities.
RESET	9	10	I	**Manual Reset:** This active high input initializes the internal logic and resets the attribute latches.
V_{CC}	40	44	I	**Power Supply:** +5 Volts ±5%
V_{BB}	1	2	I	**Bias Supply:** See figure 13.
GND	20	22	I	**Ground:** 0V reference

Product Specification

FUNCTIONAL DESCRIPTION

The VAC consists of four major sections (see block diagram). The high speed dot clock input is divided internally to provide a character clock for system timing. The parallel dot data is loaded into the video shift register on each character boundary and shifted into the video logic block at the dot rate. The six attribute inputs are latched internally and combined with the serial dot data to provide a three level video source for the monitor.

A separate BLANK input defines the active screen area. When BLANK = 0, the video levels are derived internally by the combinations of dot data, attributes, cursor, and the state of the BKGND input. Either black or white background can be selected. Symbols (dot data) are normally gray and can be highlighted to white or black as shown in figure 1.

During the inactive screen area (BLANK = 1), the video level produced by the TTL outputs is either white (BKGND = 1) or black (BKGND = 0). A separate composite blank (CBLANK) input is provided to suppress raster retrace video when white background is specified. During the inactive screen area (BLANK = 1), the video level produced by the VIDEO output is either black (BKGND = 1) or white (BKGND = 0).

For the latter case, raster retrace video suppression is accomplished by raising the BKGND input during horizontal and vertical retrace intervals. For black background, tie BKGND high. Tie CBLANK input low for both cases.

Since BLANK is delayed by 2 CCLK's internally, CBLANK must be sync'ed with the internal BLANK signal to avoid active scan data from being suppressed. A CBLANK transition must occur at least 15ns before the rising edge of DCLK in order to be latched into the 2673 (see figure 8).

Table 1. CLOCK CONTROL INPUTS

CC2	CC1	CC0	$\overline{CCLK}$ Dots/Character	$\overline{CCLK}$ Duty Cycle*
0	0	0	6	3/3
0	0	1	6	3/3
0	1	0	7	4/3
0	1	1	8	4/4
1	0	0	9	5/4
1	0	1	10	5/5
1	1	0	11	6/5
1	1	1	12	6/6

* High/low

Character Clock Counter

The character clock counter divides the frequency on the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (CC0 – CC2) as shown in table 1.

Video Shift Register

On each character boundary, the parallel data (D0 – D9) is loaded into the video shift register. The data is shifted out least significant bit first (D0) by the DCLK. If 11 or 12 dots/character are specified (CC2 – CC0 = 110 or 111), a 0 (blank dot) is always shifted out before D0. For 12 dots/character, a 0 is also shifted out after D9. The serial dot data is shifted into the video logic where it is combined with the cursor and attributes to encode three levels of video.

Attribute and Cursor Control

The VAC visual attributes capabilities include: reverse video, character blank, blink, underline, highlight, and light pen strike-thru. The six attributes and the three attribute control inputs (AMODE, AFLG, and ACD) are clocked into the VAC on the falling edge of $\overline{CCLK}$. If AFLG is high, the attributes are latched internally and are effective for either one character time (AMODE = 0) or until another set of attributes is latched (AMODE = 1). The attributes set is double buffered on a row by row basis internally. Using this technique, field attributes can extend across character row boundaries thereby eliminating the necessity of starting each row with an attribute set.

When field attribute mode is selected, (AMODE = 1), the VAC will accommodate two attribute storage configurations. In one configuration, the attribute control data is stored in the refresh RAM, taking the place of the first character code in the field to be affected. For this mode, the ACD input is tied low and blank characters will be displayed in the screen positions occupied by the attribute data (see figure 11). In the second configuration, (ACD = 1), the character codes and attribute data are presented to the VAC in parallel. In this mode, dot data is displayed at each character position (see figure 12).

The CURSOR and the attribute input signals are pipelined internally to allow for system propagations (one $\overline{CCLK}$ for refresh RAM, one $\overline{CCLK}$ for dot generator). The attribute timing signals BLINK, UL, LPL and LL are clocked into the VAC at the beginning of each scan line by the falling edge of the BLANK input. Thus, these signals must be in their proper state at the falling edge of BLANK preceding the scan line at which they are to be active (see figure 4).

Product Specification

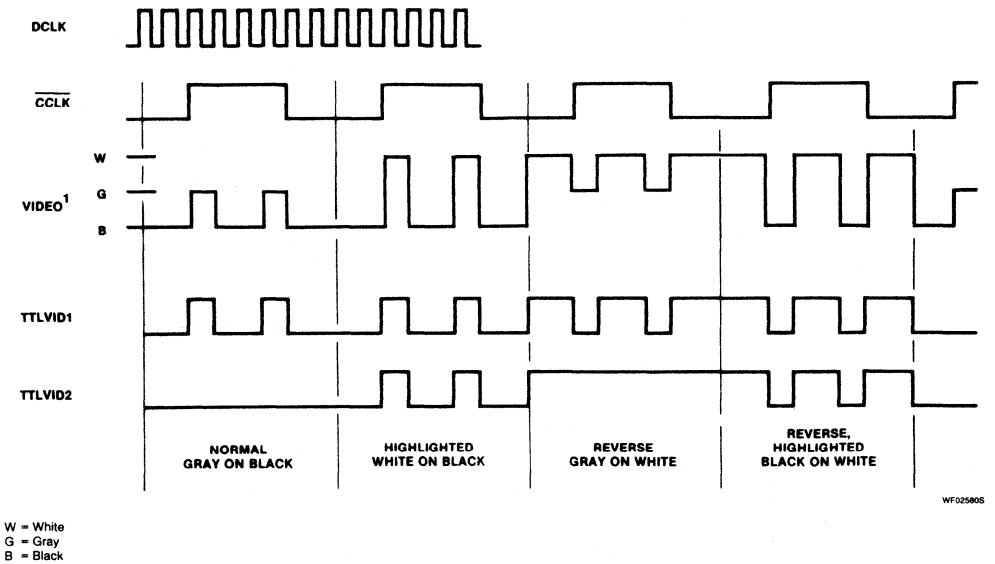


Figure 1. Encoded Video Outputs

Video Logic

The serial dot data and the pipelined cursor and attributes are combined to generate the three level current source on the VIDEO output. The three levels (white, gray, and black) are also encoded on the two TTL compatible outputs TTLVID1 and TTLVID2. The three levels are encoded as shown in table 2.

The video is normally shifted out on the leading edge of the DCLK. When the HDOT input is asserted, the corresponding dot data is delayed by one-half DCLK. This half dot shifting, when used on selected lines of character video, can be used to effect character rounding as shown in figure 2.

Attribute Hierarchy

The video of each character block consists of four components as shown in figure 3.

Table 2. VIDEO OUTPUT

TTLVID2	TTLVID1	INTENSITY
0	0	Black (or CBLANK)
0	1	Gray (on black surround)
1	0	Gray (on white surround)
1	1	White

NOTE:
The TTLVID1 output can be used independently to generate a two level non-highlighted video.

Symbol video is generated from the dot data inputs D0 – D9.

Underline video is enabled by the AUL attribute and is generated when the UL timing input is active. Underline and symbol video are always the same intensity.

Strike-thru video is enabled by the ALTPEN attribute and is generated when the LPL timing input is active. This video is always highlighted and takes precedence over the symbol and underline video. This feature applies to the 2673A only.

Surround video is the absence of symbol, underline and strike-thru video or the presence of the non-display attributes (ABLANK or ABLINK • BLINK).

The relative intensities of the four video components are determined by the remaining attributes (AHILT, ABLANK, ABLINK, ARVID) and the BKGND and CURSOR inputs as illustrated in table 3.

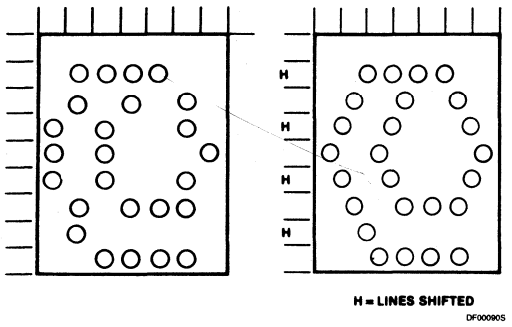


Figure 2. "AT" Symbol with and without Half Dot Shifting

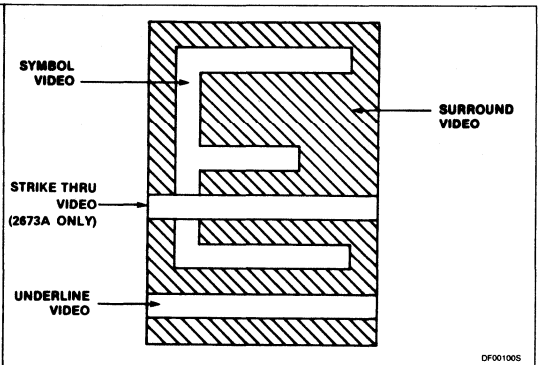


Figure 3. Video Components of Character Block

Table 3. ATTRIBUTES HIERARCHY

ATTRIBUTES AND CONTROL INPUTS d = don't care				RELATIVE VIDEO INTENSITIES W = White, B = Black, G = Gray		
BKGND[5]	Reverse[1]	Non-Display[2]	AHILT	Strike Thru Video[3]	Symbol Or Underline Video[3,4]	Surround Video[3]
0	0	0	0	W	G	B
0	0	0	1	W	W	B
0	0	1	d	B	B	B
0	1	0	0	B	G	W
0	1	0	1	B	B	W
0	1	1	d	W	W	W
1	0	0	0	B	G	W
1	0	0	1	B	B	W
1	0	1	d	W	W	W
1	1	0	0	W	G	B
1	1	0	1	W	W	B
1	1	1	d	B	B	B

NOTES:
1. Reverse = ARVID • $\overline{\text{CURSOR}}$ + $\overline{\text{ARVID}}$ • CURSOR
2. Non-display = ABLANK + ABLINK • BLINK
3. See figure 3.
4. Symbol and underline video are always the same intensity.
5. Reverse sense for VIDEO output.

Product Specification

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground	−0.5 to +6.0	V

NOTES:
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.

DC ELECTRICAL CHARACTERISTICS T_A = 0°C to +70°C, V_{CC} = 5V ±5%, V_{BB} = See figure 14[3,4,5]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage (except VIDEO)	I_{OL} = 4mA			0.4	V
V_{OH}	Output high voltage (except VIDEO)	I_{OH} = −400μA	2.4			V
V_B	VIDEO black level	R_L = 150 ohms to GND		0		V
V_G	VIDEO gray level	R_L = 150 ohms to GND		0.45		V
V_W	VIDEO white level	R_L = 150 ohms to GND		0.90		V
I_{IL}	Input low current	V_{IN} = 0.4V			−400/ −800[6]	μA
I_{IH}	Input high current	V_{IN} = 2.4V			20/40[6]	μA
I_{CC}	V_{CC} supply current	V_{IN} = 0V, V_{CC} = max			80	mA
I_{BB}	V_{BB} supply current	V_{BB} = max			120	mA

Product Specification

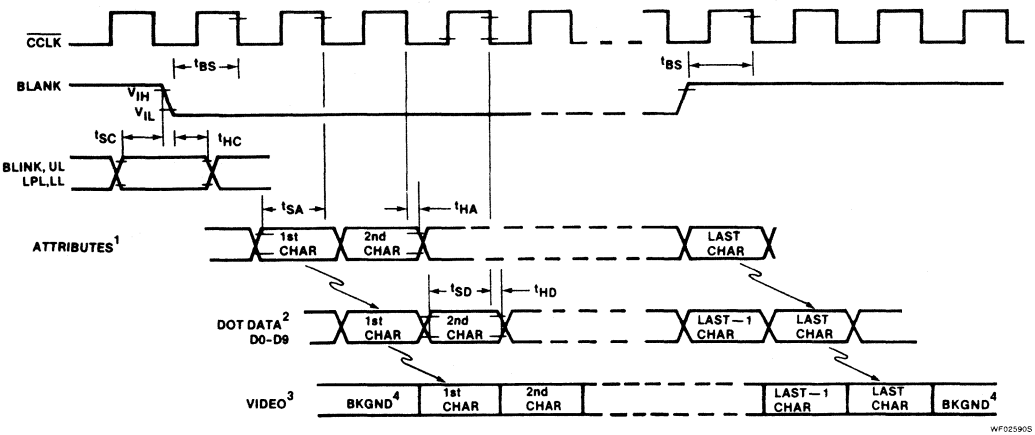
AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3,4,5]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			25MHz Version		18MHz Version		
			Min	Max	Min	Max	
Dot clock (see figure 10)							
f_d	Frequency (HDOT = 0)			25		18	MHz
	(HDOT = 1)			18		18	MHz
t_{DH}	High		15		22		ns
t_{DL}	Low		15		22		ns
Set-up times to $\overline{CCLK}$							
(See figures 4, 5, 6 and 10)							
t_{BS}	BLANK		50		50		ns
t_{SC}	BLINK, UL, LPL, LL (ref to BLANK)		20		20		ns
t_{SA}	Attributes		45		55		ns
t_{SD}	Dot data D0 – D9		70		70		ns
t_{SK}	CURSOR		50		50		ns
t_{FS}	AFLG, AMODE		50		65		ns
t_{SH}	HDOT		45		55		ns
Hold times from $\overline{CCLK}$							
(See figures 4, 5, 6 and 10)							
t_{HC}	BLINK, UL, LPL, LL (ref to BLANK)		20		20		ns
t_{HA}	Attributes		20		20		ns
t_{HD}	Dot data D0 – D9		30		30		ns
t_{HK}	CURSOR		20		20		ns
t_{FH}	AFLG, AMODE		30		30		ns
t_{HH}	HDOT		20		20		ns
Set-up times to DCLK (See figure 9)							
t_{SG}	BKGND		15		15		ns
t_{SB}	CBLANK		15		15		ns
t_{CS}	CC0 – CC2		30		35		ns
Hold times from DCLK (See figure 9)							
t_{HG}	BKGND		15		15		ns
t_{HB}	CBLANK		15		15		ns
t_{CH}	CC0 – CC2		20		20		ns
Delay times (See figures 6 and 7)		C_L = 150pF					
t_{DGM}	GMD from DCLK			65		65	ns
t_{DC}	$\overline{CCLK}$ from DCLK			65		65	ns
t_{DV}[7]	TTLVID1 and TTLVID2 from DCLK			75		80	ns
t_{DV}	VIDEO from DCLK			240		240	ns

NOTES:

3. Parameters are valid over operating temperature range unless otherwise specified.
4. All voltage measurements are referenced to ground (V_{SS}). All input signals swing between 0.4V and 2.4V. All time measurements are referenced at input voltages of 0.8V, 2.0V and at output voltages of 0.8V, 2.0V as appropriate.
5. Typical values are at +25°C, typical supply voltages and typical processing parameters.
6. For DCLK input.
7. C_L less than 150pF minimum could be faster.

Product Specification



NOTES:
1. Attributes include: ABLINK, ABLANK, ARVID, AUL, AHILT, and ALTPEN.
2. One $\overline{CCLK}$ delay for dot data (obtained from delay through character generator).
3. See figure 7 for detail timing of VIDEO, TTLVID1, TTLVID2.
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input.

Figure 4. VAC Pipeline Timing

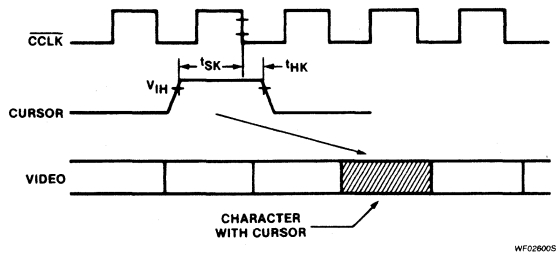
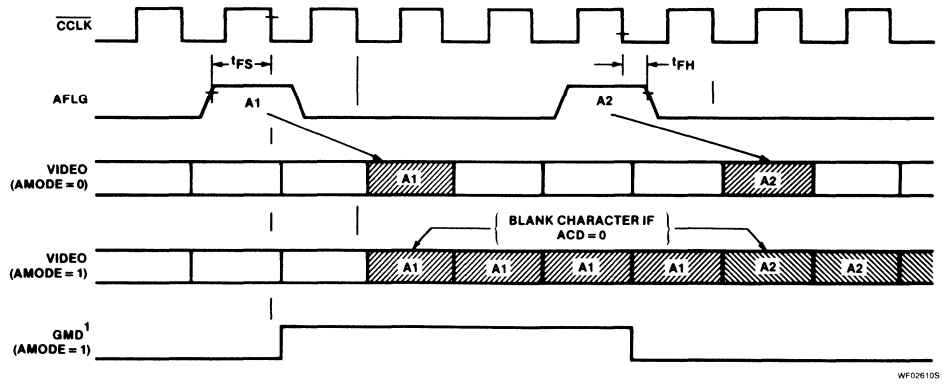
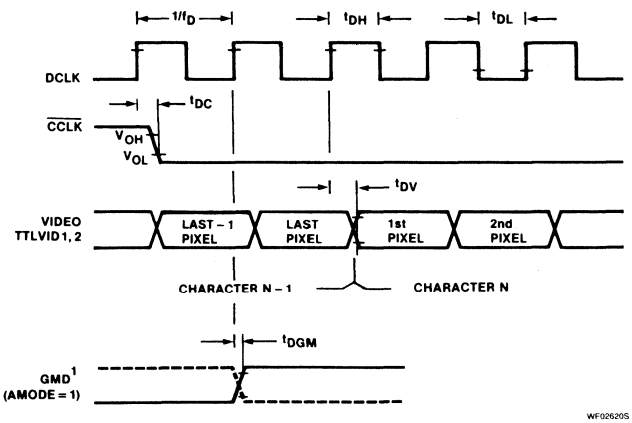


Figure 5. Cursor Pipeline Timing



NOTE:
1. GMD output in 2673B version only. See figure 7 for detail timing.

Figure 6. Character (AMODE = 0), Field (AMODE = 1), and GMD Attribute Timing



NOTE:
1. GMD output in 2673B version only.

Figure 7. Video and GMD Pipeline Timing

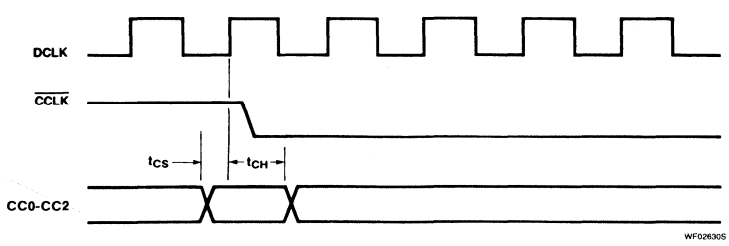


Figure 8. Character Clock Counter Timing

Product Specification

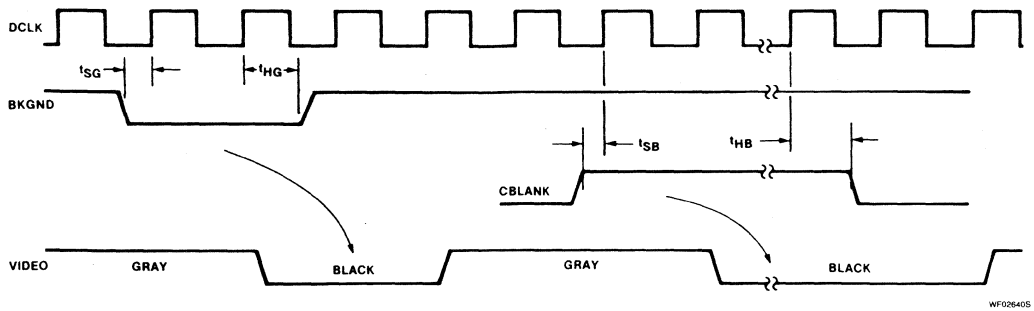
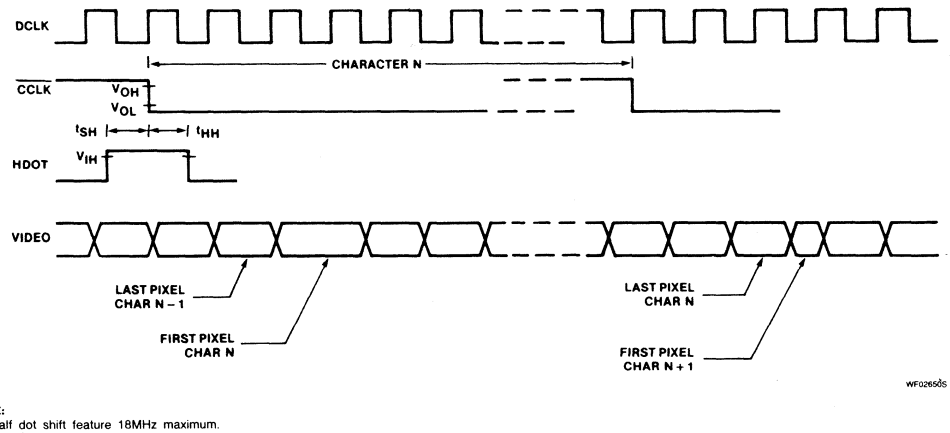


Figure 9. BKGND and RBLANK Timing During Inactive Scan Time (Blank = 1)



NOTE:
1. Half dot shift feature 18MHz maximum.

Figure 10. Half Dot Shift Timing

Product Specification

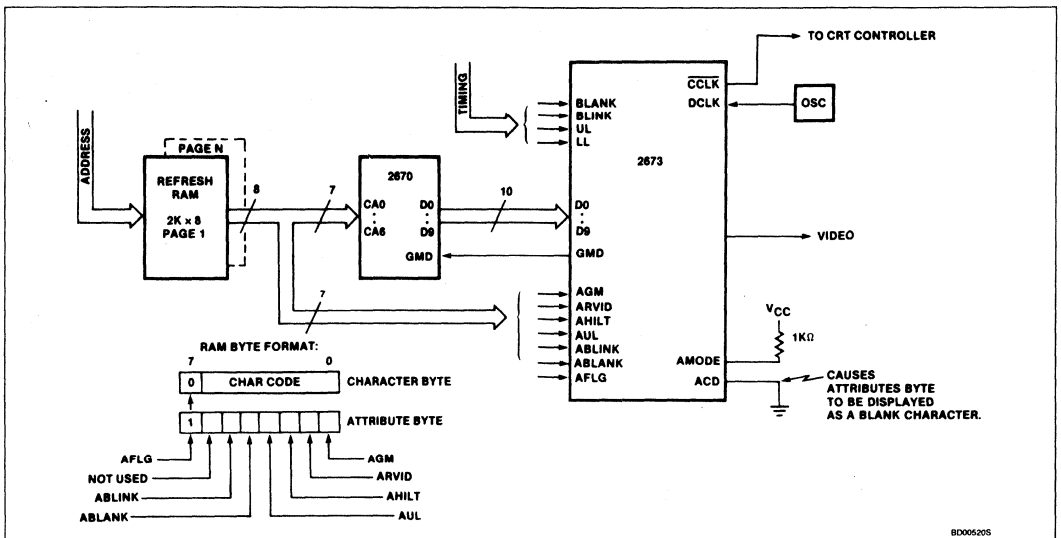
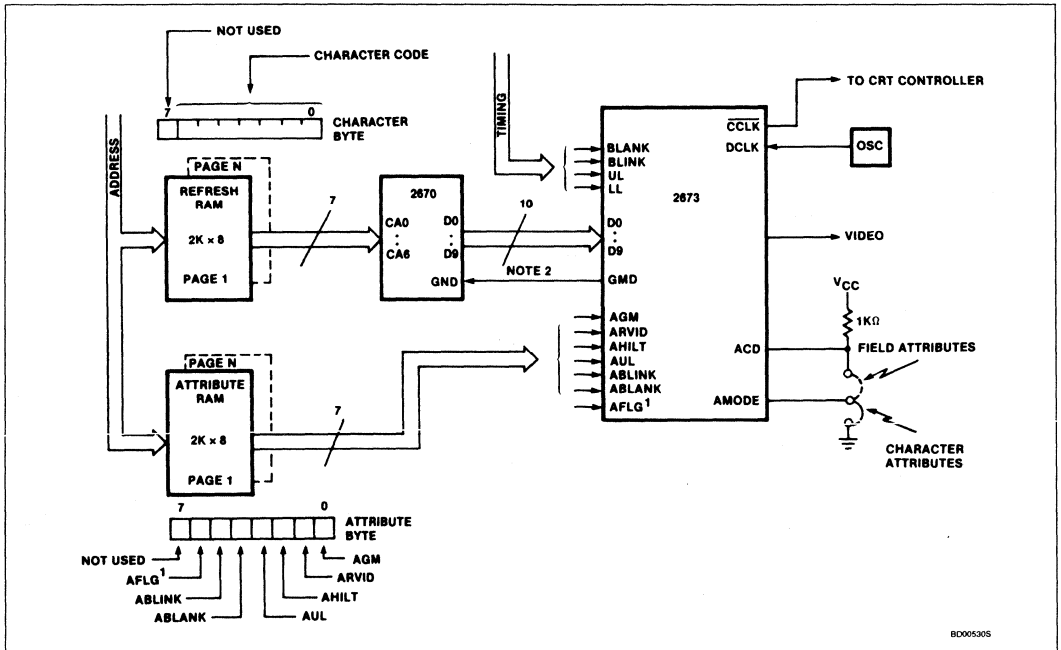


Figure 11. System Block Diagram of the 2673 in Field Attribute Mode Using the Narrow RAM (8 Wide) Configuration



NOTES:
1. For operation in character attribute mode, tie AFLG high.
2. In character attribute mode, AGM output from RAM should be connected directly to the 2670 GMD input.

Figure 12. System Block Diagram of the 2673 in Field or Character Attribute Mode Using the Wide RAM Configuration

Product Specification

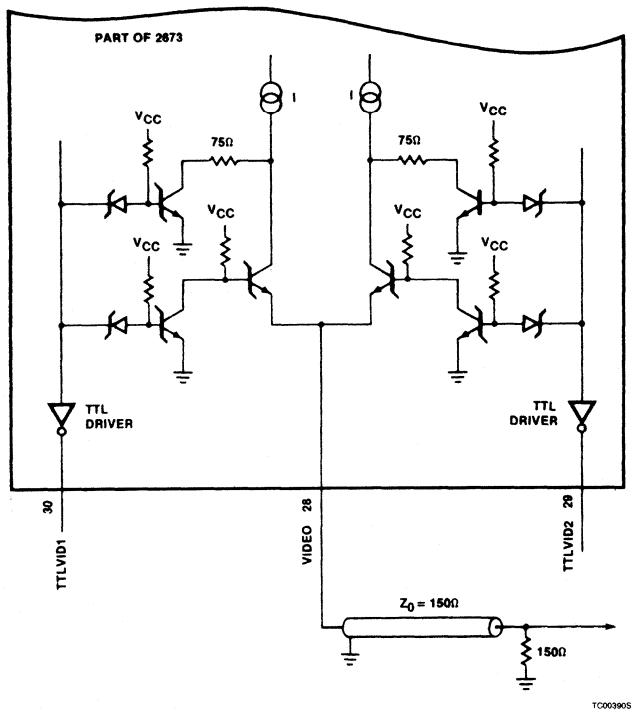


Figure 13. Video Output Stages of the 2673

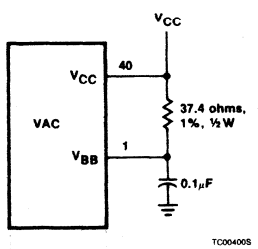


Figure 14. Test Diagram

COLOR/MONOCHROME ATTRIBUTES CONTROLLER (CMAC)

Originally published by Signetics May 1983

Preliminary

DESCRIPTION

The Signetics SCB2675 Color/Monochrome Attributes Controller (CMAC) is a bipolar LSI device designed for CRT terminals and display systems that employ raster scan techniques. It contains a programmable dot clock divider to generate a character clock, a high speed shift register to serialize input dot data into a video stream, latches and logic to apply visual attributes to the resulting display, and logic to display a cursor on the display.

The CMAC provides control of visual attributes on a character by character basis for two operating modes: monochrome and color. The monochrome mode provides reverse video, blank, highlight and two general purpose user definable attributes. In this mode, the display characters can be specified to appear on either a light or dark screen background. Retrace video suppression can be automatically or externally controlled. The color mode provides eight colors for foreground (character) video and eight colors for background video together with a luminance output for external color set selection or to simultaneously drive a monochrome monitor. Additionally, both modes provide double width, underline, blink, dot stretching and dot width attributes. In monochrome mode, the SCB2675 emulates the attribute characteristics of Digital Equipment Corporation's VT100 terminal.

The horizontal dot frequency is the basic timing input to the CMAC. This clock is divided internally to provide a character clock output for system synchronization. Up to ten bits of dot data are parallel loaded into the video shift register on each character boundary. The two TTL video data outputs in monochrome mode are encoded to provide four video intensities (black, gray, white and highlight). The video data in color mode is encoded to provide eight foreground colors and shifted out on three TTL outputs, together with the luminance output.

FEATURES

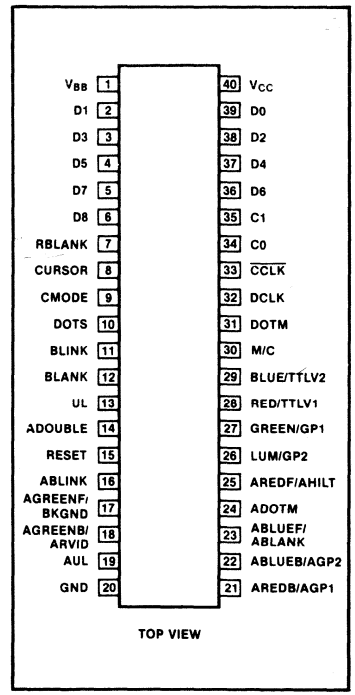
- **25 and 18MHz video dot rate versions***
- **Four video intensities encoded on two TTL outputs (monochrome mode)**
- **Eight foreground and background colors encoded on three TTL outputs (color mode)**
- **Internally latched character attributes:**
  - **— Reverse video**
  - **— Blank**
  - **— Blink**
  - **— Underline**
  - **— Highlight**
  - **— Two general purpose**
  - **— Eight foreground colors**
  - **— Eight background colors**
  - **— Dot width control**
  - **— Double width characters**
- **VT100 compatible attributes**
- **Reverse video cursor with optional white cursor in color mode**
- **Up to 10 dots per character**
- **Light or dark background in monochrome mode**
  - **— Automatic retrace blanking**
- **Programmable dot stretching**
- **Compatible with SCN2674 AVDC and SCN2670 DCGG**
- **TTL compatible**
- **40-pin dual in-line package**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**

*For faster versions consult factory.

PIN CONFIGURATION



TOP VIEW

ORDERING CODE

PACKAGES	DOTS PER CHARACTER	$V_{CC} = 5V \pm 5\%$, 0°C to +70°C	
		25MHz	18MHz
Ceramic DIP	7, 8, 9, 10	SCB2675BC5I40	SCB2675BC8I40
Plastic DIP		SCB2675BC5N40	SCB2675BC8N40
Ceramic DIP	6, 8, 9, 10	SCB2675CC5I40	SCB2675CC8I40
Plastic DIP		SCB2675CC5N40	SCB2675CC8N40

Preliminary

PIN DESIGNATION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
V_{CC}	40	I	**Power Supply:** +5VDC
V_{BB}	1	I	**Bias Supply:** See figure 5
GND	20	I	**Ground:** 0V reference
DCLK	32	I	**Dot Clock:** Dot frequency input. Video output shift rate.
$\overline{CCLK}$	33	O	**Character Clock:** An output which is a submultiple of DCLK. The period ranges from 7 to 10 DCLK periods per cycle and is determined by the state of the C0-C1 inputs.
RED/TTLV1	28	O	**Red/TTL Video 1:** In color mode, this output provides the red gun serial video. In monochrome mode, it should be used with the blue/TTL video 2 output to decode four video intensities.
BLUE/TTLV2	29	O	**Blue/TTL Video 2:** In color mode, this output provides the blue gun serial video. In monochrome mode, it should be used with the red/TTL video 1 output to decode four video intensities.
GREEN/GP1	27	O	**Green/General Purpose 1:** In color mode, this output provides the green gun serial video. In monochrome mode, it is a general purpose TTL output which is asserted if the AREDB/AGP1 input is asserted when the corresponding character dot data is loaded into the video shift register.
LUM/GP2	26	O	**Luminance/General Purpose 2:** In color mode, this output is the logical-OR of the RGB foreground video. It is low during a blanking interval and during the foreground portion of the cursor display. In monochrome mode, it is a general purpose TTL output which is asserted if the ABLUEB/AGP2 input is asserted when the corresponding character dot data is loaded into the video shift register.
UL	13	I	**Underline Timing:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK.
BLINK	11	I	**Blink Timing:** This input is sampled on the falling edge of BLANK to provide the blink rate for the blink attribute. Should be a submultiple of the frame rate.
BLANK	12	I	**Screen Blank:** When high, this input forces the video outputs to the specified background color in color mode and to the level specified by the BKGND input (either black or gray) in monochrome mode.
RBLANK	7	I	**Retrace Blank:** This input is used to force the video outputs to a low during retrace periods. If pulled high, it will automatically suppress video during the retrace periods when BLANK is high. The user may also pulse this input while BLANK is high to selectively suppress raster video.
AGREENF/BKGND	17	I	**Green Foreground/Background Intensity:** In color mode, this input activates the GREEN/GP1 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input specifies gray or black screen background.
ABLUEF/ABLANK	23	I	**Blue Foreground/Blank Attribute:** In color mode, this input activates the BLUE/TTLV2 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input generates a blank space for the associated character. The blank space intensity is controlled by the AGREENF/BKGND input, the reverse video attribute and cursor input.
AREDF/AHILT	25	I	**Red Foreground/Highlight Attribute:** In color mode, this input activates the RED/TTLV1 output during the foreground (character video) portion of the associated character block. In monochrome mode, this input highlights the associated character (including underline).
CURSOR	8	I	**Cursor Timing:** This input provides the timing for the cursor video. In color mode, with CURSOR and CMODE high, the RGB outputs are driven high (white cursor). If CMODE is low, or in monochrome mode, this input reverses the intensities of the video and attributes. Cursor position, shape, and blink rate are controlled by this input.
CMODE	9	I	**Cursor Mode:** Used in color mode only. When CURSOR and CMODE are high, the RGB outputs are driven high (white cursor). When CURSOR is high and CMODE is low, the RGB outputs are logically inverted (reverse video cursor).
AUL	19	I	**Underline Attribute:** Specifies a line to be displayed in the character block. The specific line(s) are specified by the UL input. All other attributes apply to the underline video.

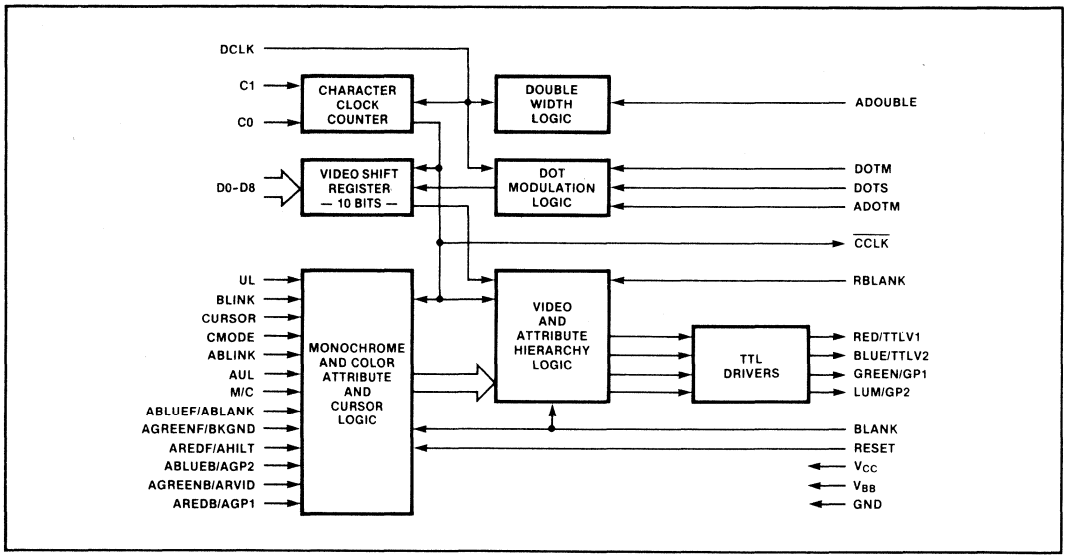
Preliminary

PIN DESIGNATION (Continued)

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
ABLINK	16	I	**Blink Attribute:** In color mode, this active high input will drive the foreground RGB combination to the background RGB combination. In monochrome mode, the associated character or background is driven to the intensity determined by BKGND, reverse video attribute and the cursor input.
ADOUBLE	14	I	**Double Width Attribute:** This active high input causes the associated character video to be shifted out of the serial shift register at one half the dot frequency (DCLK). The $\overline{CCLK}$ output is not affected.
AREDB/AGP1	21	I	**Red Background/General Purpose Attribute 1:** In color mode, this input activates the RED/TTLV1 output during the background portion of the associated character block. In monochrome mode, it activates the GREEN/GP1 output for the associated character block.
ABLUEB/AGP2	22	I	**Blue Background/General Purpose Attribute 2:** In color mode, this input activates the BLUE/TTLV2 output during the background portion of the associated character block. In monochrome mode, it activates the LUM/GP2 output for the associated character block.
AGREENB/ARVID	18	I	**Green Background/Reverse Video Attribute:** In color mode, this input activates the GREEN/GP1 output during the background portion of the associated character block. In monochrome mode, it causes the associated character block video intensities to be reversed.
D0-D8	36-39, 2-6	I	**Dot Data Input:** These are parallel inputs corresponding to the character/graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the trailing (falling) edge of each character clock ($\overline{CCLK}$).
C0-C1	34-35	I	**Character Clock Control:** The states of these two static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate.
RESET	15	I	**Reset:** This active high input initializes the internal logic and resets the attribute latches.
M/C	30	I	**Monochrome/Color Mode:** This input selects whether the CMAC operates in monochrome or color mode. A low selects color mode and a high selects monochrome mode.
ADOTM	24	I	**Dot Modulation Attribute:** When DOTM and this input are high, the active dot width of the associated character video is one DCLK. When DOTM is high and this input is low, the active dot width of the associated character video is two DCLKs.
DOTM	31	I	**Dot Width Modulation:** When this input is high, two DCLKs are used for each dot shifted through the shift register. When this input is low, one DCLK is used.
DOTS	10	I	**Dot Stretching:** Sampled at the falling edge of BLANK. When this input is high, one extra dot is appended to individual dots or groups of dots of the input parallel data and then transferred through the shift register. When this input is low, normal transfer of input parallel data results.

Preliminary

BLOCK DIAGRAM



FUNCTIONAL DESCRIPTION

The CMAC consists of seven major sections (see block diagram). The high speed dot clock input is applied to a programmable divider to provide a character clock output for system timing. Parallel dot data is loaded into the video shift register on character boundaries and shifted into the video logic block at the dot rate specified by the dot modulation section. The appropriate attribute control inputs are selected by the mode select logic, latched internally on character boundaries, and combined with the serial dot data to provide monochrome or color video outputs.

The BLANK input defines the active screen area. In color mode, the video outputs are forced to the specified background color when this signal is asserted; in monochrome mode the video outputs are forced to the states defined by the BKGND input, i.e., black if dark background is selected and gray if light background is selected. A separate RBLANK input allows the user to select the amount of border around the active area when operating in color mode or in monochrome mode with light background. This input can be tied high, in which case the area outside the active area will be dark, or it may be pulsed during BLANK periods to externally control the border widths.

In color mode, eight colors for the character (foreground) and eight colors for the background (area other than character) can be selected by the attribute inputs. In monochrome mode, the intensities of foreground and background are a function of the attribute and BKGND inputs, i.e., characters may be black, gray, white, or highlight (very white) while background may be black, gray, or white (see Table 1).

Table 1 MONOCHROME MODE ATTRIBUTE CHARACTERISTICS

REV[1]	AHILT	ABLINK[2]	FOREGROUND VIDEO	BACKGROUND VIDEO
0	0	0	W	B
0	0	1	W/G	B
0	1	0	H	B
0	1	1	H/W	B
1	0	0	B	G
1	0	1	B/W	G/B
1	1	0	B	W
1	1	1	B/H	W/B

NOTES

1. REV = (BKGND) XOR (ARVID):

BKGND	ARVID	REV
0	0	0
0	1	1
1	0	1
1	1	0

2. For blinking, the video outputs are shown as 0/1, where 0 and 1 are the blink timing input states.
3. Foreground includes underline when underlining is specified by AUL = 1.
4. When ABLANK = 1, foreground component becomes same as background component.
5. Codes for video outputs are as follows:

CODE	TTLV2	TTLV1	BEAM INTENSITY
B	0	0	Black
G	0	1	Gray
W	1	0	White
H	1	1	Highlight

Preliminary

Character Clock Counter

The character clock counter divides the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (C1-C0) as follows:

C1	C0	SCB2675B	
		DOTS/ CHAR.	$\overline{CCLK}$ DUTY CYCLE*
0	0	10	5/5
0	1	7	4/3
1	0	8	4/4
1	1	9	5/4

*High/low

C1	C0	SCB2675C	
		DOTS/ CHAR.	$\overline{CCLK}$ DUTY CYCLE*
0	0	10	5/5
0	1	6	3/3
1	0	8	4/4
1	1	9	5/4

*High/low

The number of dot clocks/character is normally the number of dots/character as listed above. However, when dot width control is specified, the DCLK input is divided by two before it is applied to the character clock counter resulting in the number of dot clocks/character being double those listed above, although the number of displayed dots/character remains the same. See Dot Modulation section of this data sheet.

Video Shift Register

On each character boundary, the parallel input dot data (D0-D8) is loaded into the video shift register. The data is shifted out least significant bit first (D0) at the DCLK rate. If 10 dots/character are specified (C1-C0 = 00), the tenth dot will be the same as D8. The serial dot data from the video shift register is routed to the video logic where it is combined with the cursor and attribute control bits to produce the video data outputs.

Mode Select, Attribute and Cursor Control

The mode select logic multiplexes the monochrome and color attribute inputs and outputs as specified by the M/C input. The monochrome mode provides blank, reverse video, highlight and two general purpose attributes. The latter may be used, with external logic, to combine other attributes (e.g., overscore) into the video stream. The color mode provides RGB foreground and background color attributes. Both modes provide double width characters, blink, underline, dot width control and dot stretching.

The cursor and attribute inputs are pipelined internally to allow for system pipeline propagations. The cursor input signal is delayed internally by two $\overline{CCLK}$s (one for RAM and one for the character generator), while the attribute inputs are delayed for one $\overline{CCLK}$ to account for the delay of the character data through the character generator latches. The attribute timing inputs (BLINK, UL and DOTS) are clocked into the 2675 at the beginning of each scan line time by the falling edge of BLANK. Thus, these inputs must be in their proper state at the falling edge of BLANK preceding the scan line where they are required to be active. The BLANK signal itself is also delayed internally to provide for the RAM and character generator delays (see figures 6 and 7). Internal delays cause the video outputs to be delayed relative to $\overline{CCLK}$ as illustrated in figure 8.

Video Logic

Each character block consists of the three components shown in figure 1. Symbol video is generated from the dot data inputs D0-D8. Underline video is enabled by the AUL attribute and is generated during the scan lines for which the UL input is active. Underline and symbol video are always the same intensity or color, and other attributes (e.g., ABLINK) apply to them equally. The combination of underline and symbol video is also referred to as foreground video. Background video is the area of the character block corresponding to the absence of foreground video. The assertion of the non-display attribute (ABLANK) causes the entire character block to be displayed as background.

In monochrome mode, the serial dot data and pipelined cursor and attributes are combined to generate four video intensities (black, gray, white and highlight) which are encoded on the TTLV1 and TTLV2 outputs as follows:

TTLV2	TTLV1	VIDEO INTENSITY
0	0	Black
0	1	Gray
1	0	White
1	1	Highlight

Table 1 describes the relationship between attributes and video intensity of the foreground and background components of the character block in monochrome mode.

In color mode, the colors of the foreground and background components are specified by the corresponding attribute inputs; AREDF, AGREENF and ABLUEF dictate the color of the foreground component while AREDB, AGREENB and ABLUEB do the same for the background component. In this mode, the serial dot data and pipelined cursor and attributes are combined to generate four video outputs. The RED, GREEN and BLUE outputs separately contain the corresponding foreground and background components. The LUM output is the logical-OR of the foreground colors and can be used to drive a separate monochrome monitor or to select a different set of colors for the foreground.

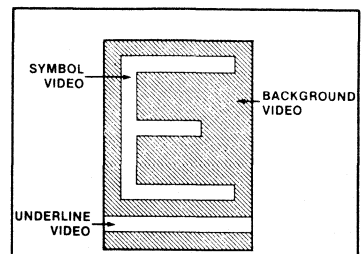


Figure 1. Character Block Definition

Preliminary

Dot Modulation Logic

The dot modulation logic controls the video shift register to supply dot stretching and dot width control.

Dot stretching is controlled by the DOTS input which is sampled each scan line at the trailing (falling) edge of BLANK. If DOTS is asserted at that time, all characters on the following scan line will have dot stretching applied. Dot stretching causes an extra dot to be added to individual dots or groups of dots as shown in figures 2 and 3. Dot stretching can be used to:

1. Compensate for low video bandwidth monitors (since the minimum active displayed segment with dot stretching is two DCLKs).
2. Assure crisp black characters when operating in white background mode.
3. Provide thick characters as a means of distinguishing areas of the display.

Dot width is controlled by the DOTM and ADOTM inputs. DOTM is tied either high, which enables the feature on the entire display, or low, which disables the feature. With ADOTM high, the dot width of characters can be selectively controlled by assertion of the ADOTM attribute input. When operating in this mode, the dot clock input is divided by two before being applied to other circuits in the CMAC. This affects the $\overline{CCLK}$ output.

When dot width control is enabled as above, two DCLKs are used for each video dot period. Asserting ADOTM for a particular character will cause each active video dot of the displayed character to be turned on for one DCLK and off for the other DCLK, while if ADOTM is negated for that character, the active video dot for that character will be turned on (black background) or off (white background) for both DCLK times (see figures 2 and 4). Only the character video component of the character block is modulated. Underline video and background are not affected by on-time modulation. Width control can be used to:

1. Make horizontal lines and vertical lines appear the same brightness on the display.
2. Provide two different brightness levels for characters without requiring a monitor with analog brightness inputs.

However, note that the effects produced by this feature are highly dependent on the video amplifier characteristics of the monitor used.

Double Width Logic

The double width logic controls the rate at which dots are shifted through the video shift register. When the ADOUBLE input is asserted, the associated character video will be shifted at one half the DCLK rate, and the dot information for the next character will be loaded into the shift register two $\overline{CCLK}$s later. The $\overline{CCLK}$ output is not affected. If a double width character is specified at the last location of a character row, the second half of the double width character (one $\overline{CCLK}$) will extend into the horizontal front porch.

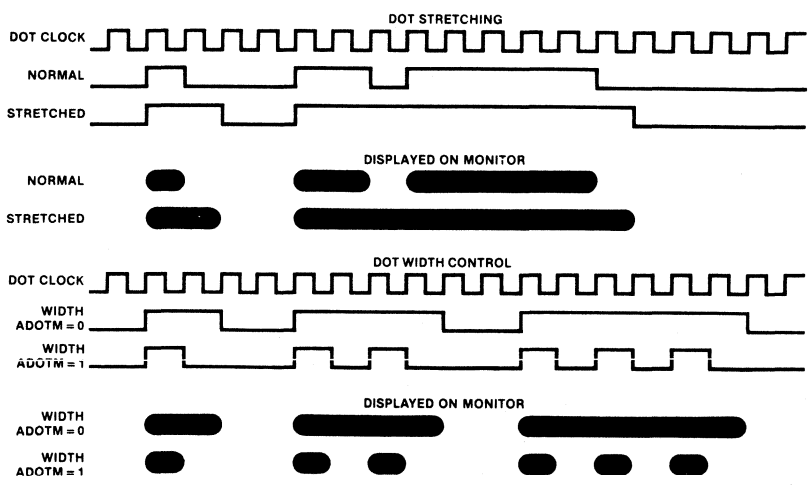


Figure 2. Dot Modulation Timing

Preliminary

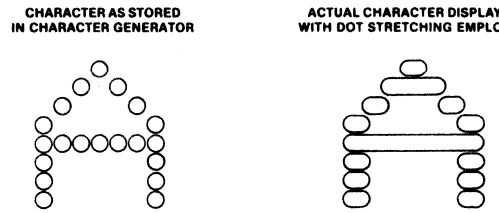


Figure 3. Dot Stretching

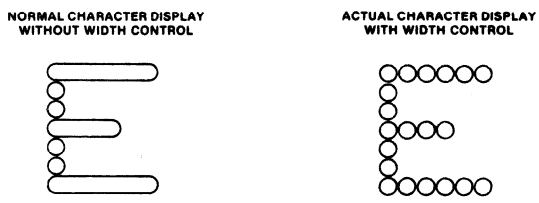


Figure 4. Dot Width Control

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = figure 5[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS Min	LIMITS Typ	LIMITS Max	UNIT
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 4mA$			0.4	V
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input low current	$V_{IN} = 0.4V$				
	DCLK				−800	μA
	All other inputs				−400	μA
I_{IH}	Input high current	$V_{IN} = 2.4V$				
	DCLK				40	μA
	All other inputs				20	μA
I_{CC}	V_{CC} supply current	$V_{IN} = 0V$, V_{CC} = max			80	mA
I_{BB}	V_{BB} supply current	Figure 5			120	mA

Preliminary

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = figure 5[4,5,6]

PARAMETER	TEST CONDITIONS	TENTATIVE LIMITS				UNIT
		25MHz VERSION		18MHz VERSION		
		Min	Max	Min	Max	
Dot clock timing[7]						
f_D Frequency			25		18	MHz
t_{DH} High time		15		22		ns
t_{DL} Low time		15		22		ns
Setup times[8]						
t_{SB} BLANK to $\overline{CCLK}$		40		50		ns
t_{SA} Attributes to $\overline{CCLK}$		40		50		ns
t_{SD} D0-D9 to $\overline{CCLK}$		60		70		ns
t_{SK} CURSOR to $\overline{CCLK}$		40		50		ns
t_{SC} C0, C1 to DCLK		20		20		ns
t_{SR} RBLANK to DCLK		20		20		ns
t_{SM} BLINK, UL, DOTS to BLANK		20		20		ns
Hold times[8]						
t_{HB} BLANK from $\overline{CCLK}$		20		20		ns
t_{HA} Attributes from $\overline{CCLK}$		20		20		ns
t_{HD} D0-D8 from $\overline{CCLK}$		30		30		ns
t_{HK} CURSOR from $\overline{CCLK}$		20		20		ns
t_{HC} C0, C1 from DCLK		20		20		ns
t_{HR} RBLANK from DCLK		20		20		ns
t_{HM} BLINK, UL, DOTS from BLANK		20		20		ns
Delay times[7]	$C_L = 50pF$					
t_{DC} $\overline{CCLK}$ from DCLK			55		70	ns
t_{DV} Other outputs from DCLK		30	60	35	70	ns

NOTES
1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions other than those described in the AC and DC Electrical Characteristics section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 3ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and at output voltages of 0.8V and 2.0V as appropriate.
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. See figure 8.
8. See figures 6, 7, 9, and 10.

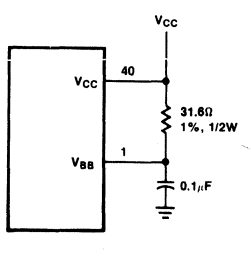
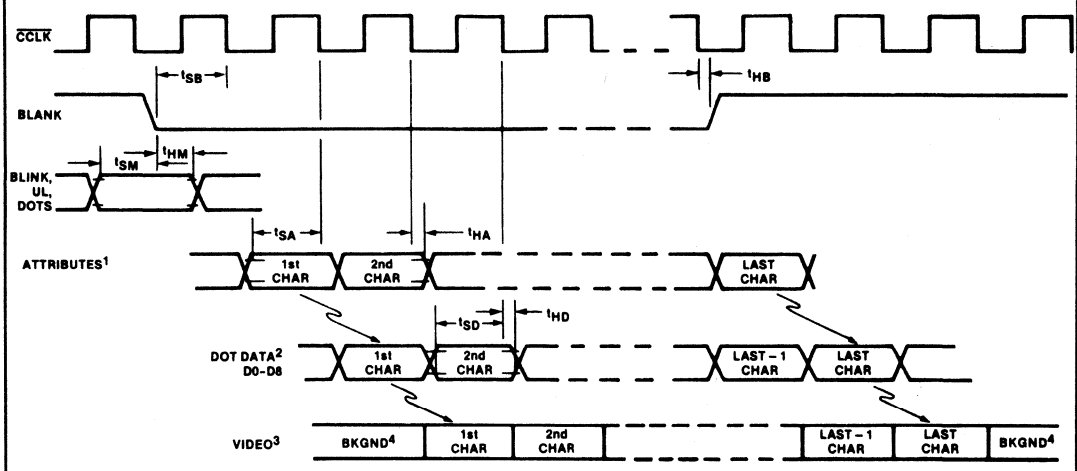


Figure 5.
Recommended V_{BB} Test Circuit

Preliminary



NOTES:
1. Attributes include: ABLINK, ABLANK, ARVID, AUL, AHILT, ADOUBLE, ADOTM, two general purpose, and foreground/background colors.
2. One $\overline{CCLK}$ delay for dot data (obtained from delay through character generator).
3. See figure 8 for detail timing of video outputs.
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input in monochrome mode.

Figure 6. CMAC Pipeline Timing

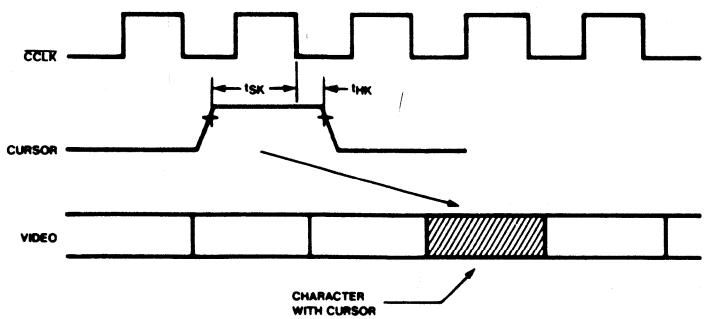


Figure 7. Cursor Pipeline Timing

Preliminary

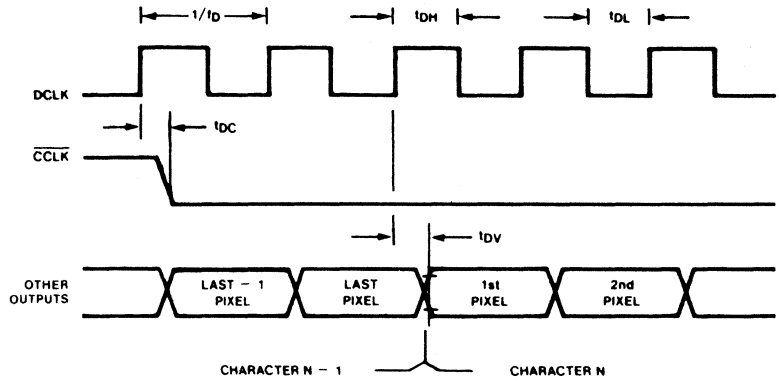


Figure 8. Output Pipeline Timing

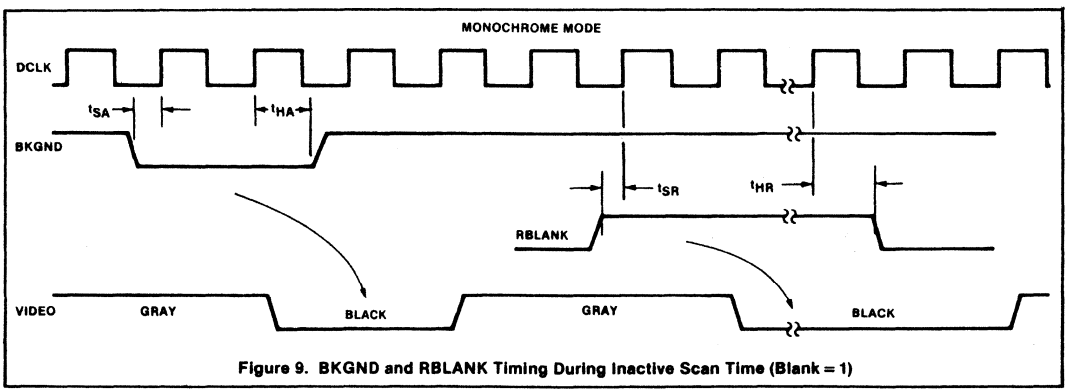
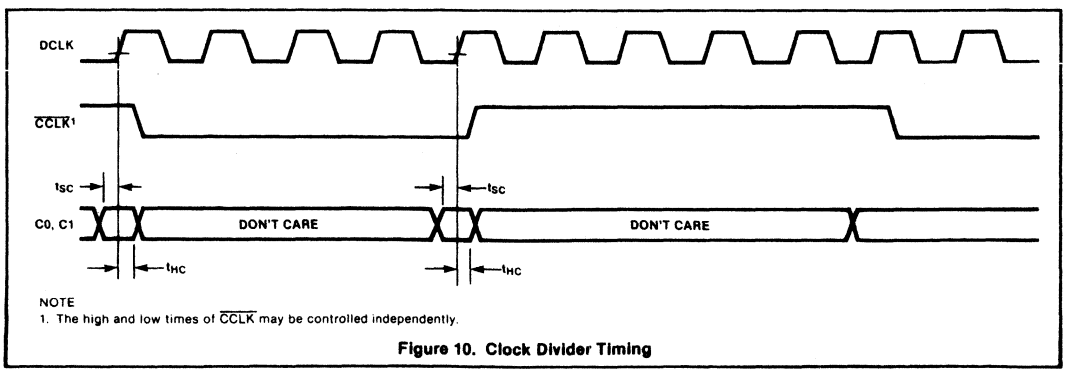


Figure 9. BKGND and RBLANK Timing During Inactive Scan Time (Blank = 1)



NOTE
1. The high and low times of $\overline{CCLK}$ may be controlled independently.

Figure 10. Clock Divider Timing

Preliminary

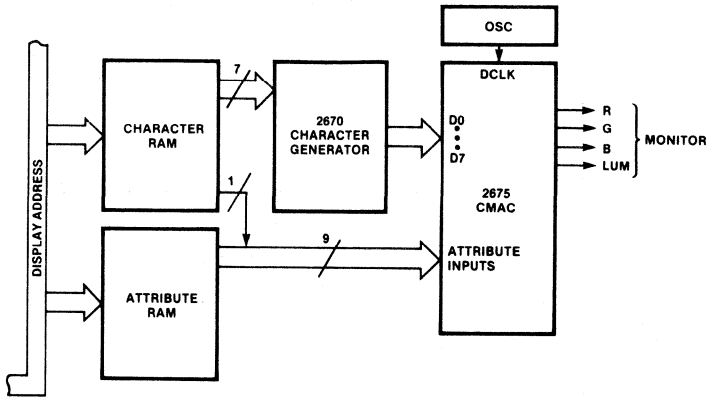


Figure 11. System Block Diagram of SCB2675 in Color Mode

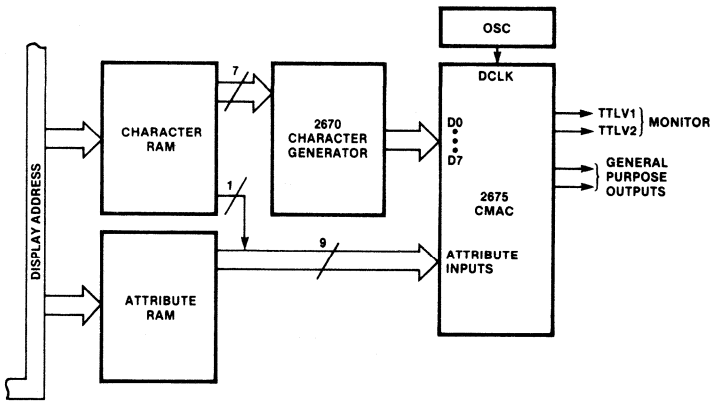


Figure 12. System Block Diagram of SCB2675 in Monochrome Mode

Video Attributes Controller (VAC)

Originally published by Signetics February 1985

Product Specification

DESCRIPTION

The Signetics SCB2677A and SCB-2677B Video Attributes Controllers (VAC) are bipolar LSI devices designed for CRT terminals and display systems that employ raster scan techniques. Each contains a high speed video shift register, field and character attributes logic, attribute latch, cursor format logic and half dot shift control.

The VAC provides control of visual attributes on a field or character by character. Internal logic preserves field attribute data from character row to character row so that an attribute byte is not required at the beginning of each row. The SCB2677B provides for reverse video, blank (non-display), blink, underline and highlight attributes and a graphics mode attribute to work in conjunction with the Signetics SCN2670 Display Character and Graphics Generators (DCGG). The SCB2677A substitutes a strike-thru attribute for the graphics attribute.

The horizontal dot frequency is the basic timing input to the VAC. Internally, this clock is divided down to provide a character clock output for system synchronization. Up to ten bits of video dot data are parallel loaded into the video shift register on each character boundary. The video data is encoded to three levels of intensity (black, gray and white) and output on two TTL outputs. Light or dark screen background may be specified.

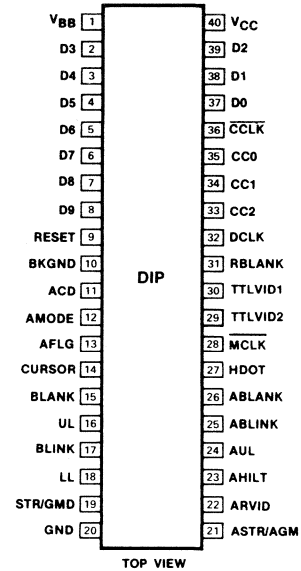
FEATURES

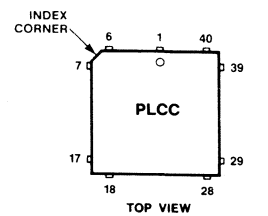
- **18MHz and 25MHz video dot rate**
- **Three level encoded TTL video outputs**
- **Character/field attribute logic**
  - **Reverse video**
  - **Character blank**
  - **Character blink**
  - **Underline**
  - **Highlight**
- **Strike-thru or graphics control**
- **Field attributes extend from row to row**
- **Light or dark field**
- **Cursor reverse video logic**
- **Up to 12 dots per character**
- **Retrace blanking for light field**
- **Optional field graphics control output**
- **High speed bipolar design**
- **TTL compatible**
- **Compatible with Signetics SCN2672 PVTC, SCN2674 AVDC and SCN2670 DCGG**
- **Upgrade of the Signetics SCB2673 VAC**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**

PIN CONFIGURATION



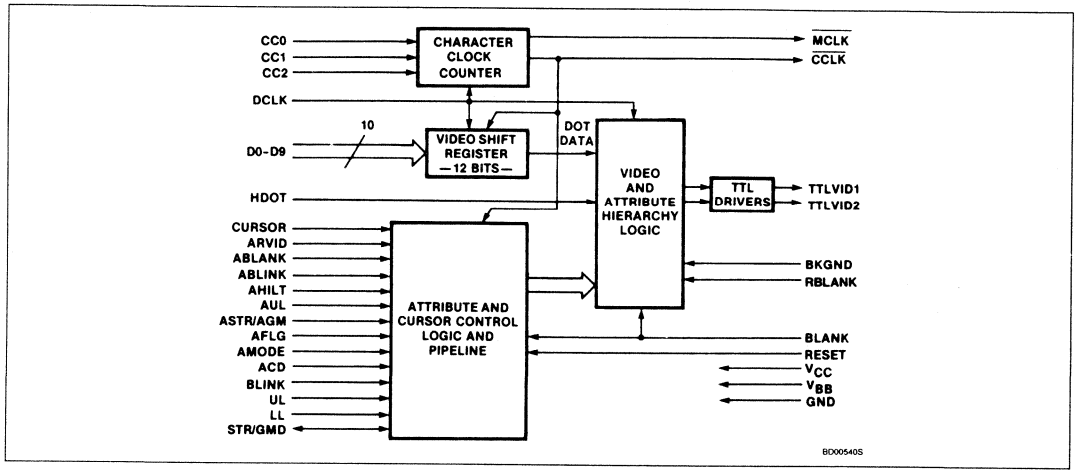


Pin	Function	Pin	Function	Pin	Function
1	NC	16	CURSOR	30	HDOT
2	V_{BB}	17	BLANK	31	$\overline{MCLK}$
3	D3	18	UL	32	TTLVID2
4	D4	19	BLINK	33	TTLVID1
5	D5	20	LL	34	NC
6	D6	21	STR/GMD	35	RBLANK
7	D7	22	GND	36	DCLK
8	D8	23	NC	37	CC2
9	D9	24	ASTR/AGM	38	CC1
10	RESET	25	ARVID	39	CC0
11	BKGND	26	AHILT	40	$\overline{CCLK}$
12	NC	27	AUL	41	D0
13	ACD	28	ABLINK	42	D1
14	AMODE	29	ABLANK	43	D2
15	AFLG			44	V_{CC}

ORDERING CODE

PACKAGES	V_{CC} = 5V± 5%, T_A = 0°C to 70°C			
	Graphics Attribute		Strike-Thru Attribute	
	25MHz	18MHz	25MHz	18MHz
Ceramic DIP	SCB2677BC5I40	SCB2677BC8I40	SCB2677AC5I40	SCB2677AC8I40
Plastic DIP	SCB2677BC5N40	SCB2677BC8N40	SCB2677AC5N40	SCB2677AC8N40
Plastic LCC	SCB2677BC5A44	SCB2677BC8A44	SCB2677AC5A44	SCB2677AC8A44

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
DCLK	32	36	I	**Dot Clock:** Dot frequency input. Video output shift rate.
$\overline{CCLK}$	36	40	O	**Character Clock:** A submultiple of DCLK. The frequency ranges from one sixth to one twelfth of DCLK, as determined by the state of the CC0–CC2 inputs.
CC2 – CC0	33 – 35	37 – 39	I	**Character Clock Control:** The logic state on these three static inputs determine the internal divide factor for the $\overline{CCLK}$ output rate. Character clock rates of 6 through 12 dots per character may be specified.
D0 – D9	37 – 39, 2 – 8	41 – 43, 3 – 9	I	**Dot Data Input:** These are parallel inputs corresponding to the character/graphic symbol dot data for a given scan line. These inputs are strobed into the video shift register on the falling edge of each character clock.
HDOT	27	30	I	**Half Dot Shift:** When this input is high, the serial video output is delayed by one half dot time. This input is latched on the falling edge of each character clock.
CURSOR	14	16	I	**Cursor Timing:** This input provides the timing for the cursor video. When high, effectively reverses the intensities of the video and attributes. Cursor position, shape, and blink rate are controlled by this input.
BKGND	10	11	I	**Background Intensity:** Specifies light or dark video during BLANK and character fields. Affects the intensities of all attributes.
BLANK	15	17	I	**Screen Blank:** When high, this input forces the video outputs to the level specified by the BKGND input (either high or low intensity). Not effective when RBLANK is high.
RBLANK	31	35	I	**Retrace Blank:** This input is used to force the two video outputs to a low intensity (black) during retrace intervals. If held high (1), it will automatically suppress video when BLANK is high (1). The user may pulse this input while BLANK is high to selectively suppress raster video.
ARVID	22	25	I	**Reverse Video Attribute:** The intensity of the associated character or field video is reversed. All other attributes are effectively reversed.
AHILT	23	26	I	**Highlight Attribute:** All dot video (including underline) of the associated character or field is highlighted with respect to the BKGND input and the reverse video attribute.
ABLANK	26	29	I	**Blank Attribute:** Generates a blank space in the associated character or field. The blank space intensity is determined by the BKGND input, the reverse video attribute, and the CURSOR input.
ABLINK	25	28	I	**Blink Attribute:** The associated character or field video is driven to the intensity determined by BKGND and the reverse video attribute when the BLINK input is high.
AUL	24	27	I	**Underline Attribute:** Specifies a line to be displayed on the character or field. The line is specified by the UL input. All other attributes apply to the underline video.
ASTR/AGM	21	24	I	**Strike-Thru Attribute (2677A):** Specifies a line to be displayed on the character or field. The line is specified by the STR input. **Attribute Graphics Mode (2677B):** This input is latched and synchronized to provide a field GMD output for the SCN2670 DCGG.
AMODE	12	14	I	**Attribute Mode:** Specifies character (AMODE = 0) or field (AMODE = 1) attribute mode.
AFLG	13	15	I	**Attributes Flag:** The VAC samples and latches the attributes inputs when this input is high. If field attributes are specified (AMODE = 1), the attributes are double buffered on a row basis. Thus, each scan line of every character row will start with the attributes that were valid at the end of the previous row.
ACD	11	13	I	**Attribute Control Display:** In field attributes mode (AMODE = 1), if ACD = 0, the first character in each new attribute field (the attribute control character) will be suppressed and only the attributes will be displayed. If ACD = 1, the first character and the attributes are displayed. This input has no effect in character mode (AMODE = 0).
BLINK	17	19	I	**Blink:** This input is sampled on the falling edge of the BLANK to provide the blink rate for the character blink attribute. It should be a submultiple of the frame rate.
UL	16	18	I	**Underline:** Indicates the scan line(s) for the underline attribute. Latched on the falling edge of BLANK.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO. DIP	PIN NO. PLCC	TYPE	NAME AND FUNCTION
STR/GMD	19	21	I O	**Strike-Thru Line (2677A):** Indicates the scan line(s) for the strike-thru attribute. Latched on the falling edge of BLANK. **Graphics Mode (2677B):** This output provides a synchronized, latched, field graphics mode corresponding to the AGM input. This output can be used to control the GM input on the SCN2670 DCGG.
LL	18	20	I	**Last Line:** Indicates the last scan line of each character row. Used internally to extend field attributes across row boundaries. Latched on the falling edge of BLANK. This input has no effect in character mode (AMODE = 0).
$\overline{MCLK}$	28	31	O	**Memory Clock:** This output is active for the last dot time for each $\overline{CCLK}$ period. See figure 1.
TTLVID1	30	33	O	**TTL Video 1:** This output corresponds to the serial, non-highlighted video dot pattern.
TTLVID2	29	32	O	**TTL Video 2:** This output corresponds to the highlighted serial video dot pattern. Should be used with TTLVID1 to decode a composite video of three intensities.
RESET	9	10	I	**Manual Reset:** This active high input initializes the internal logic and resets the attribute latches.
V_{CC}	40	44	I	**Power Supply:** +5 volts.
V_{BB}	1	2	I	**Bias Supply:** See figure 14.
GND	20	22	I	**Ground:** 0V reference.

FUNCTIONAL DESCRIPTION

The VAC consists of four major sections (see block diagram). The high speed dot clock input is divided internally to provide a character clock for system timing. The parallel dot data is loaded into the video shift register on each character boundary and shifted into the video logic block at the dot rate. The six attribute inputs are latched internally and combined with the serial dot data to provide a three level video source for the monitor.

A separate BLANK input defines the active screen area. When BLANK = 0, the video levels are derived internally by the combinations of dot data, attributes, cursor, and the state of the BKGND input. Either black, gray or white background can be selected. Symbols (dot data) are normally gray and can be highlighted to white or black as shown in figure 2.

During the inactive screen area (BLANK = 1), the video level produced by the TTL outputs is either gray ($\overline{BKGND}$ = 1) or black (BKGND = 0). A separate retrace blank (RBLANK) input is provided to suppress raster retrace video when gray background is specified. This input will force the video outputs to a low if RBLANK and BLANK = 1. The user may pulse RBLANK during the retrace interval in order to extend the gray border closer to the monitor edges.

Character Clock Counter

The character clock counter divides the frequency on the DCLK input to generate the character clock ($\overline{CCLK}$). The divide factor is specified by the clock control inputs (CC0 – CC2) as shown in the table above. See figure 1.

CC2	CC1	CC0	$\overline{CCLK}$ Dots/Character	$\overline{CCLK}$ Duty Cycle*
0	0	0	6	3/3
0	0	1	6	3/3
0	1	0	7	4/3
0	1	1	8	4/4
1	0	0	9	5/4
1	0	1	10	5/5
1	1	0	11	6/5
1	1	1	12	6/6

*Low/high

Video Shift Register

On each character boundary, the parallel data (D0 – D9) is loaded into the video shift register. The data is shifted out least significant bit first (D0) by the DCLK. If 11 or 12 dots/character are specified (CC2 – CC0 = 110 or 111), a 0 (blank dot) is always shifted out before D0. For 12 dots/character, a 0 is also shifted out after D9. The serial dot data is shifted into the video logic where it is combined with the cursor and attributes to encode three levels of video.

Attribute and Cursor Control

The VAC visual attributes capabilities include: reverse video, character blank, blink, underline, highlight, and strike-thru. The six attributes and the three attribute control inputs (AMODE, AFLG, and ACD) are clocked into the VAC on the falling edge of $\overline{CCLK}$. If AFLG is high, the attributes are latched internally and are effective for either one character time (AMODE = 0) or until another set of attributes is latched (AMODE = 1). The attributes set is double buffered on a row by row basis internally. Using this technique, field attributes can extend across character row boundaries thereby eliminating the necessity of starting each row with an attribute set.

When field attribute mode is selected, (AMODE = 1), the VAC will accommodate two attribute storage configurations. In one configuration, the attribute control data is stored in the refresh RAM, taking the place of the first character code in the field to be affected. For this mode, the ACD input is tied low and blank characters will be displayed in the screen positions occupied by the attribute data (see figure 12). The display RAM contains intermixed character and attribute data. When new attribute data is written to the SCB2677, the AFLG input is set high. The character at that location will be blanked, and only the attribute information will be dis-

played. That particular attribute data will be used for the resultant characters until the next AFLG pulse occurs. In the second configuration (ACD = 1), the character codes and attribute data are presented to the VAC in parallel (i.e., there are separate RAMs for the character and attribute data). In this mode, dot data is displayed at each character position (see figure 13).

The CURSOR and the attribute input signals are pipelined internally to allow for system propagations (one $\overline{CCLK}$ for refresh RAM, one $\overline{CCLK}$ for dot generator). The attribute timing signals BLINK, UL, STR and LL are clocked into the VAC at the beginning of each scan line by the falling edge of the BLANK input. Thus, these signals must be in their proper state at the falling edge of BLANK preceding the scan line at which they are to be active (see figure 5). The SCN2670 DCGG delays the character dot data by one character clock. The VAC assumes that there is a DCGG in the system and latches the dot data one character clock later than the latching of the attribute data.

Video Logic

The serial dot data and the pipelined cursor and attributes are combined to generate three levels (white, gray, and black) on two TTL compatible outputs, TTLVID1 and TTLVID2. The three levels are encoded as shown to the right.

The video is normally shifted out on the leading edge of the DCLK. When the HDOT input is asserted, the corresponding dot data is delayed by one-half DCLK. This half dot shifting, when used on selected lines of character video, can be used to effect eye-pleasing character rounding as shown in figure 3. The half dot shift does not extend into the next character's field boundary.

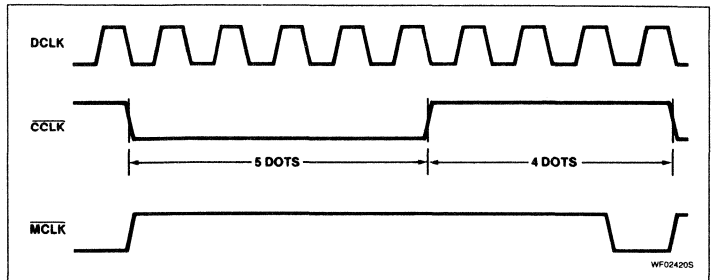


Figure 1. Clock Waveform (9 Dots/Character)

TTLVID2	TTLVID1	INTENSITY
0	0	Black
0	1	Gray
1	0	Not used
1	1	White

NOTE:
The TTLVID1 output can be used independently to generate a two level non-highlighted video.

Attribute Hierarchy

The video of each character block consists of four components as shown in figure 4.

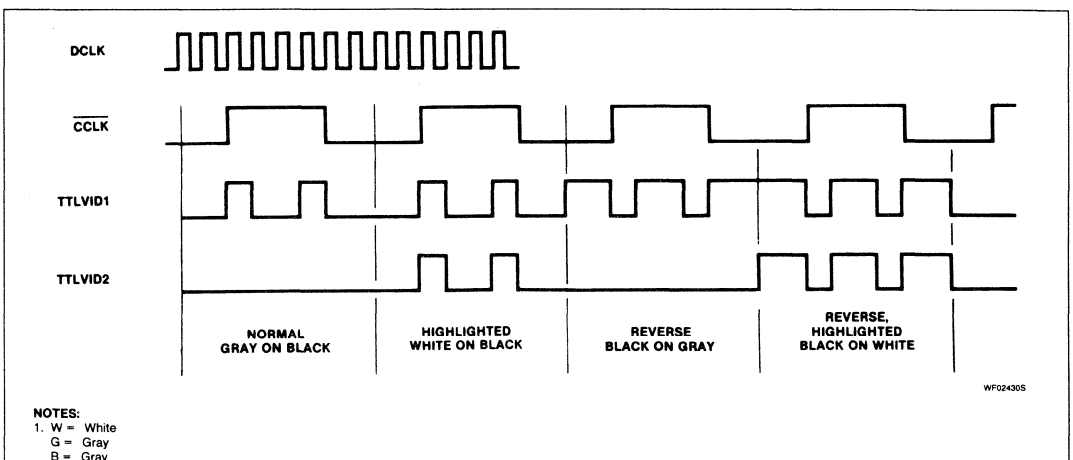
Symbol video is generated from the dot data inputs D0 – D9.

Underline video is enabled by the AUL attribute and is generated when the UL timing input is active. Underline and symbol video are always the same intensity.

Strike-thru video is enabled by the ASTR attribute and is generated when the STR timing input is active. This video is the same intensity as the symbol and underline video. This feature applies to the SCB2677A only.

Surround video is the absence of symbol, underline and strike-thru video or the presence of the non-display attributes (ABLANK or ABLINK • BLINK).

The relative intensities of the four video components are determined by the remaining attributes (AHILT, ABLANK, ABLINK, ARVID) and the BKGND and CURSOR inputs as illustrated in table 1.



NOTES:
1. W = White
   G = Gray
   B = Gray

Figure 2. Encoded Video Outputs

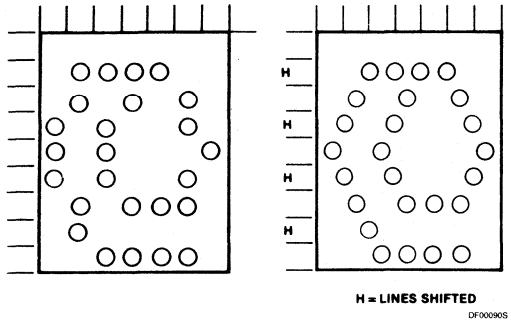


Figure 3. "AT" Symbol with and without Half Dot Shifting

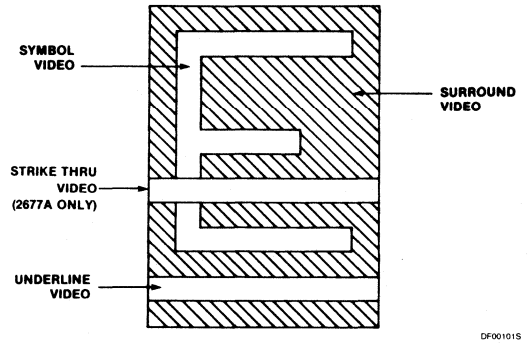


Figure 4. Video Components of Character Block

Table 1. ATTRIBUTES HIERARCHY

BLANK	RBLANK	BKGND	REVERSE[1]	AHILT	"NON-" DISPLAY[2]	SYMBOL, UNDERLINE OR STRIKE-THRU[3,4]	SURROUND VIDEO[3]
0	d	0	0	0	0	G	B
0	d	0	0	0	1	B	B
0	d	0	0	1	0	W	B
0	d	0	0	1	1	B	B
0	d	0	1	0	0	B	G
0	d	0	1	0	1	G	G
0	d	0	1	1	0	B	W
0	d	0	1	1	1	W	W
0	d	1	0	0	0	B	G
0	d	1	0	0	1	G	G
0	d	1	0	1	0	B	W
0	d	1	0	1	1	W	W
0	d	1	1	0	0	G	B
0	d	1	1	0	1	B	B
0	d	1	1	1	0	W	B
0	d	1	1	1	1	B	B
1	0	0	d	d	d	B	B
1	0	1	d	d	d	G	G
1	1	d	d	d	d	B	B

NOTES:
B = Black
G = Gray
W = White
d = Don't care
1. REVERSE = ARVID ⊕ CURSOR
2. Non-display = (ABLINK • BLINK) + ABLANK
3. See figure 4.
4. Symbol, underline and strike-thru are always same intensity.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3,4,5,6]

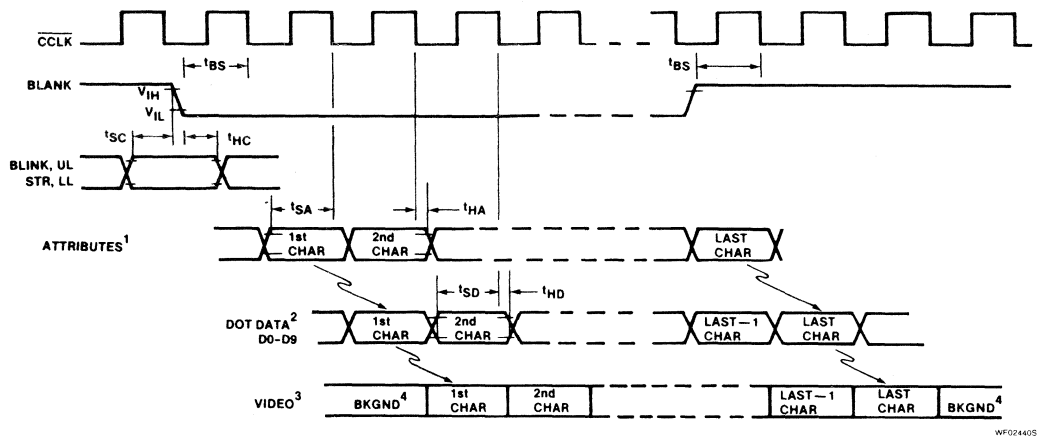
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 4mA$			0.4	V
V_{OH}	Output high voltage	$I_{OH} = -400\mu A$	2.4			V
I_{IL}	Input low current	V_{IN}=0.4V			−400/ −800[6]	μA
I_{IH}	Input high current	V_{IN} = 2.4V			20/40[6]	μA
I_{CC}	V_{CC} supply current	V_{IN} = 0V, V_{CC} = max			80	mA
I_{BB}	V_{BB} supply current	V_{BB} = max			120	mA

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. Operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. Parameters are valid over operating temperature range unless otherwise specified.
4. All voltage measurements are referenced to ground (V_{SS}). All input signals swing between 0.4V and 2.4V. All time measurements are referenced at input voltages of 0.8V, 2.0V and at output voltage of 0.8V, 2.0V as appropriate.
5. Typical values are at +25°C, typical supply voltages and typical processing parameters.
6. For DCLK input.
7. C_L less than 150pF minimum could be faster.

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, V_{BB} = See figure 14[3, 4, 5, 6]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			25MHz Version		18MHz Version		
			Min	Max	Min	Max	
Dot clock (figure 11)							
f_D	Frequency			25		18	MHz
t_{DH}	High		15		22		ns
t_{DL}	Low		15		22		ns
Set-up times to $\overline{CCLK}$ (figures 5, 6, 7 and 11)							
t_{BS}	BLANK		50		50		ns
t_{SC}	BLINK, UL, STR, LL (ref to BLANK)		20		20		ns
t_{SA}	Attributes		45		55		ns
t_{SD}	Dot data D0 – D9		70		70		ns
t_{SK}	CURSOR		50		50		ns
t_{FS}	AFLG		50		65		ns
t_{SH}	HDOT		45		55		ns
Hold times from $\overline{CCLK}$ (figures 5, 6, 7 and 11)							
t_{HC}	BLINK, UL, STR, LL (ref to BLANK)		20		20		ns
t_{HA}	Attributes		20		20		ns
t_{HD}	Dot data D0 – D9		30		30		ns
t_{HK}	CURSOR		20		20		ns
t_{FH}	AFLG		30		30		ns
t_{HH}	HDOT		20		20		ns
Set-up times to DCLK (figures 9,10)							
t_{SG}	BKGND		15		15		ns
t_{SB}	RBLANK		15		15		ns
t_{CS}	CC0 – CC2		30		35		ns
Hold times from DCLK (figures 9,10)							
t_{HG}	BKGND		15		15		ns
t_{HB}	RBLANK		15		15		ns
t_{CH}	CC0 – CC2		20		20		ns
Delay times (figures 7 and 8)		C_L = 150pF					
t_{DGM}	GMD from DCLK			65		65	ns
t_{DC}	$\overline{MCLK}$, $\overline{CCLK}$ from DCLK			65		65	ns
t_{DV}[7]	TTLVID1 and TTLVID2 from DCLK			75		80	ns



NOTES:
1. Attributes include: ABLINK, ABLANK, ARVID, AUL, AHILT, and ASTR.
2. One $\overline{CCLK}$ delay for dot date (obtained from delay through character generator).
3. See figure 8 for detail timing of TTLVID1,TTLVID2
4. Non-active scan time. VIDEO reverts to polarity selected by the BKGND input.

Figure 5. VAC Pipeline Timing

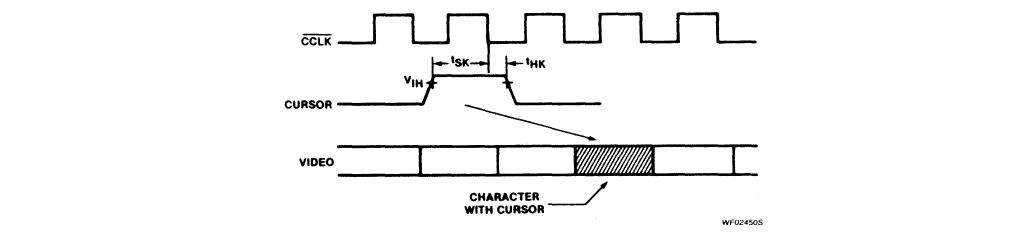
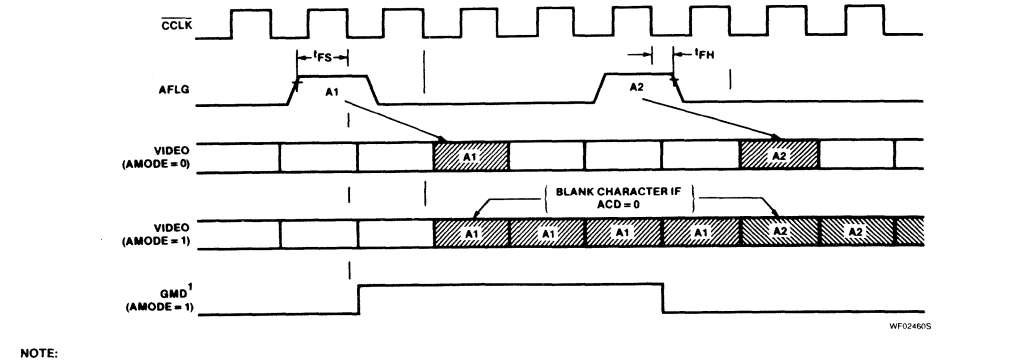
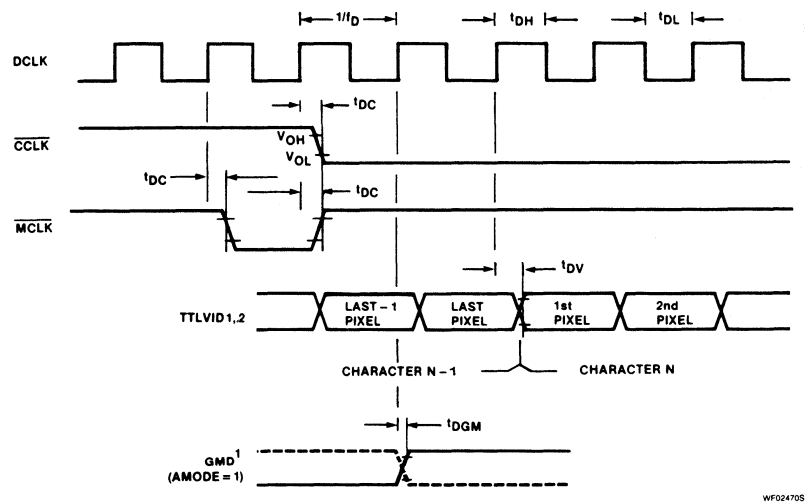


Figure 6. Cursor Pipeline Timing



NOTE:
1. GMD output in SCB2677B version only. See figure 8 for detail timing.

Figure 7. Character (AMODE = 0), Field (AMODE = 1), and GMD Attribute Timing



NOTE:
1. GMD output in SCB2677B version only.

Figure 8. Video and GMD Pipeline Timing

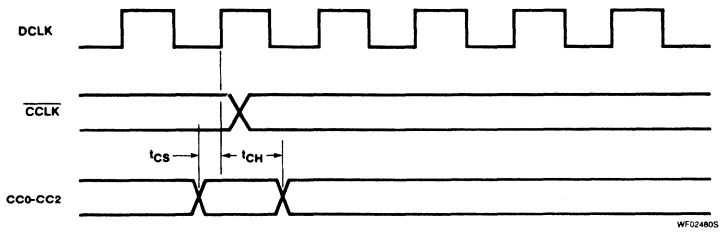


Figure 9. Character Clock Counter Timing

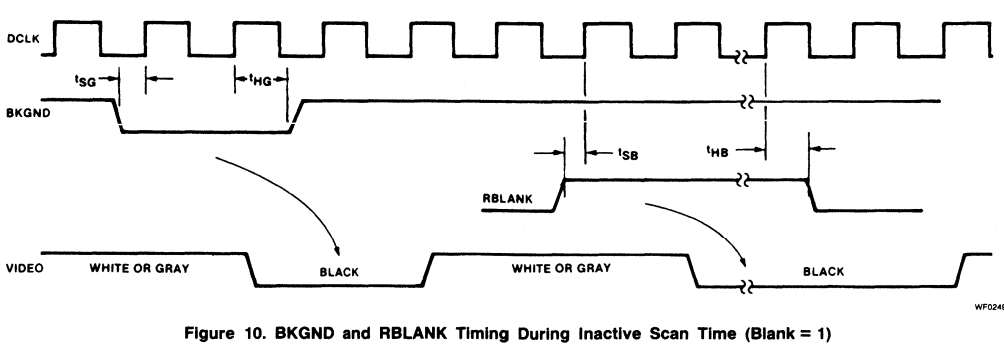


Figure 10. BKGND and RBLANK Timing During Inactive Scan Time (Blank = 1)

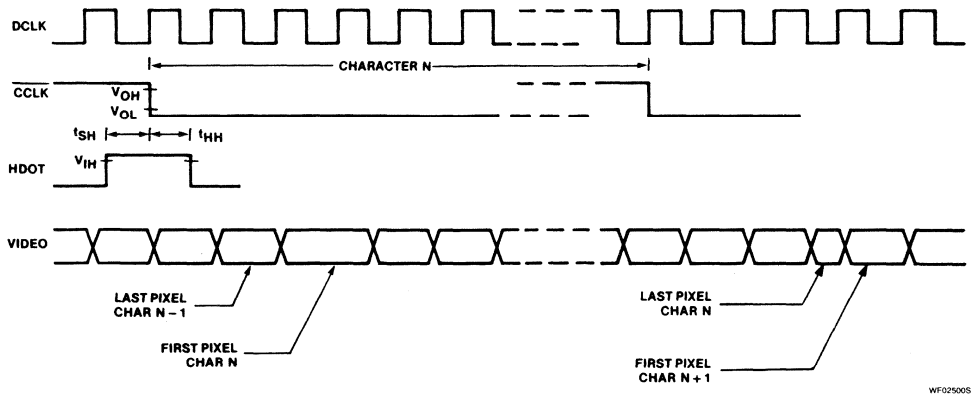


Figure 11. Half Dot Shift Timing

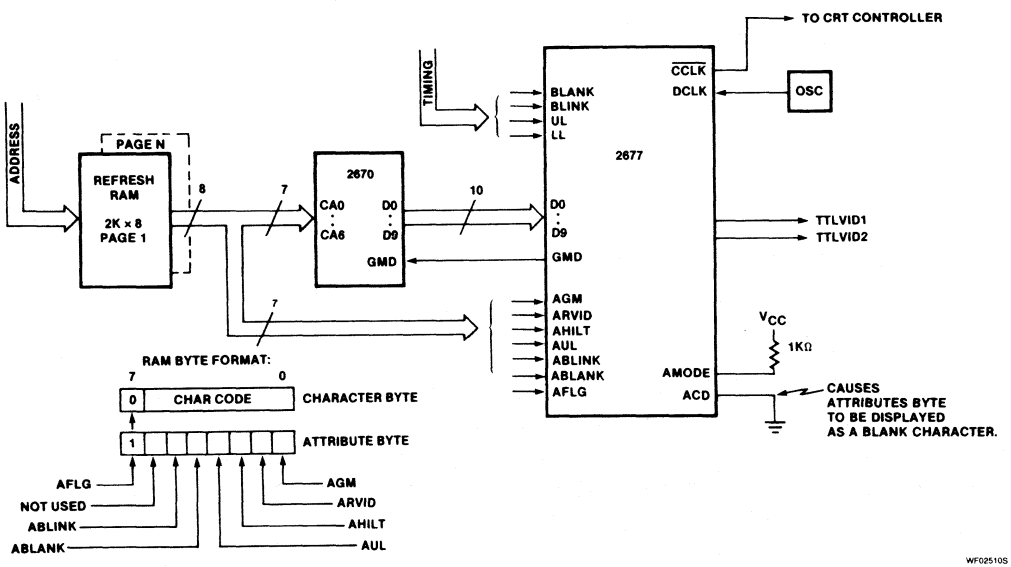
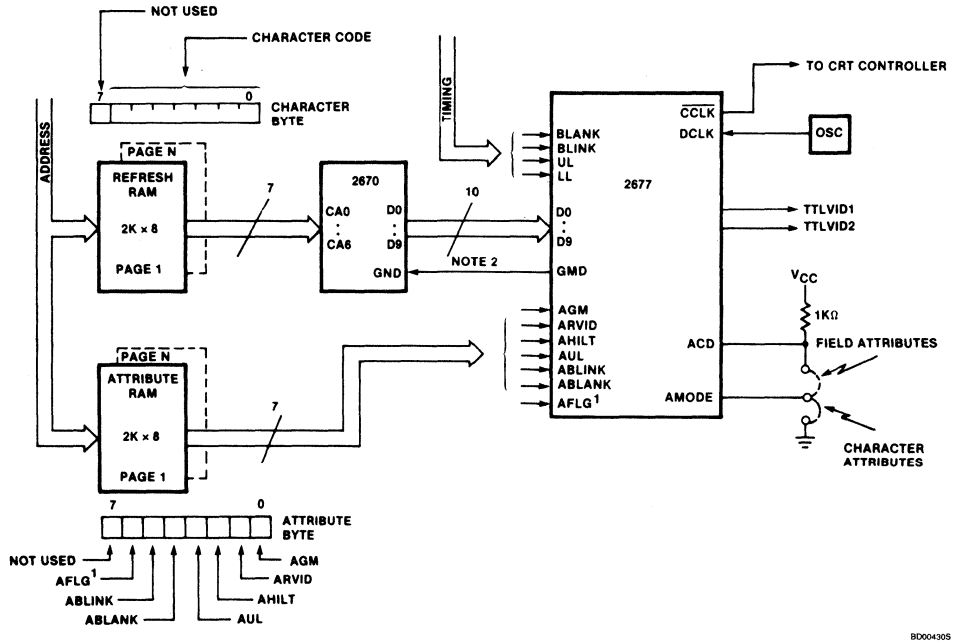


Figure 12. System Block Diagram of the 2677 in Field Attribute Mode Using the Narrow RAM (8 Wide) Configuration



NOTE:
1. For operation in character attribute mode, tie AFLG high.
2. In character attribute mode, AGM output from RAM should be connected directly to the SCN2670 GMD input.

Figure 13. System Block Diagram of the 2677 in Field or Character Attribute Mode Using the Wide RAM Configuration

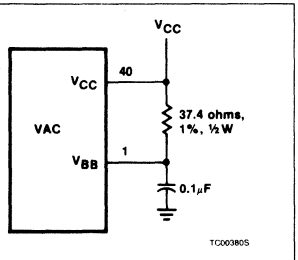


Figure 14. Test Diagram

Display Character and Graphics Generator (DCGG)

Product Specification

Originally published by Signetics February 1985

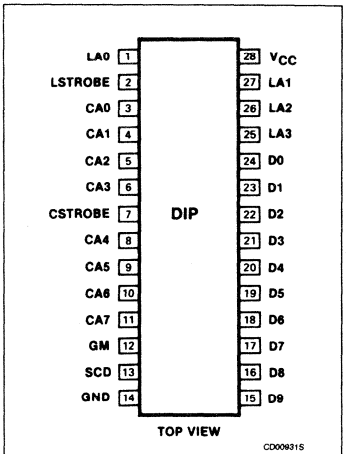
DESCRIPTION

The Signetics SCN2670 Display Character and Graphics Generator (DCGG) is a mask-programmable 11,648-bit line select character generator. It contains 128 10 x 9 characters placed in a 10 x 16 matrix, and has the capability of shifting certain characters, such as j, y, g, p and q, that normally extend below the baseline. Character shifting, previously requiring additional external circuitry, is now accomplished internally by the DCGG; effectively, the 9 active lines are lowered within the matrix to compensate for the character's position.

FEATURES

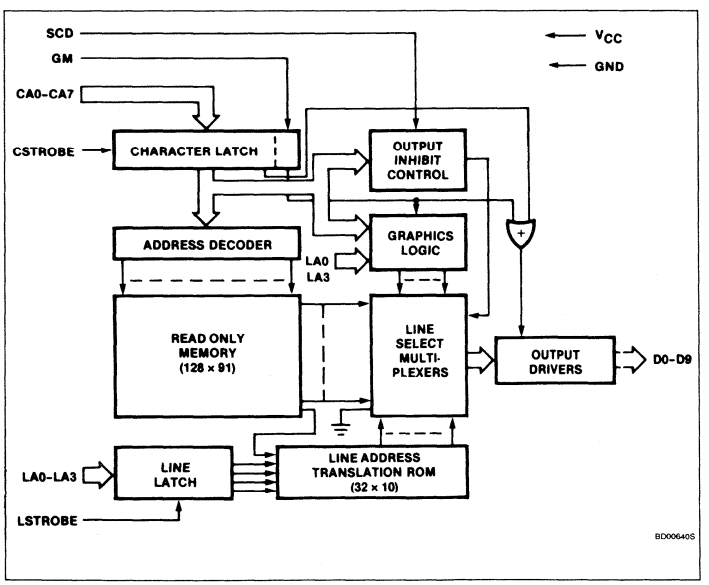
- **5 x 7 or 7 x 9 character fonts**
- **128 10 x 9 matrix characters**
- **256 graphic characters**
- **Optional thin graphics for forms**
- **Character and line address latches**
- **Internal descend logic**
- **200nsec and 300nsec character select access time versions**
- **Control character output inhibit logic**
- **Static operation — no clocks required**
- **Single 5V power supply**
- **TTL compatible inputs and outputs**

PIN CONFIGURATION



NOTE:
Pin functions same as DIP.

BLOCK DIAGRAM



ORDERING CODE

PACKAGES	CHARACTER FONT SIZE	V_{CC} = 5V ±5%, T_A = 0°C to 70°C	
		t_{CA} = 200ns	t_{CA} = 300ns
Ceramic DIP Plastic DIP Plastic LCC	5 X 7	SCN2670CC2I28 SCN2670CC2N28 SCN2670CC2A28	SCN2670CC3I28 SCN2670CC3N28 SCN2670CC3A28
Ceramic DIP Plastic DIP Plastic LCC	7 X 9	SCN2670BC2I28 SCN2670BC2N28 SCN2670BC2A28	SCN2670BC3I28 SCN2670BC3N28 SCN2670BC3A28

PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
CA0 – CA7	3 – 6, 8 – 11	I	**Character Address:** Eight bit code specifies the character or graphic pattern for which matrix data is to be supplied. In character mode (GM = 0), CA0 through CA6 select one of the 128 ROM-defined characters and CA7 is a chip enable. The outputs are active when CA7 = 1 and are three-stated when CA7 = 0. In graphics mode (GM = 1), the outputs are always active and CA0 through CA7 select one of 256 possible graphic patterns to be output.
CSTROBE	7	I	**Character Strobe:** Used to store the character address (CA0 through CA7) and graphics mode (GM) inputs into the character latch. Data is latched on the negative going edge of CSTROBE. Can be connected to the $\overline{CCLK}$ of the Signetics attributes controller chips.
GM	12	I	**Graphics Mode:** GM = 0 (low) selects character mode; GM = 1 (high) selects graphics mode.
LA0 – LA3	1, 25 – 27	I	**Line Address:** In character mode, selects one of the 16 lines of matrix data for the selected character to appear at the 10 outputs. LA0 is the LSB and LA3 is the MSB. The input codes which cause each of the nine lines of character data to be output are specified as part of the programming data for both non-shifted and shifted fonts. Cycling through the nine specified counts at the LA0 through LA3 inputs cause successive lines of data to be output on D0 through D9. The 7 non-specified codes for both non-shifted and shifted characters cause blanks (logic zeros) to be output. In graphics mode, the line address gates the latched graphics data directly to the outputs.
LSTROBE	2	I	**Line Strobe:** Used to store the line address data (LA0 through LA3) in the line address latch. Data is latched on the negative going edge of LSTROBE. Can be connected to the BLANK output of the Signetics video display controller chips.
SCD	13	I	**Selected Character Disable:** In character mode, a high level at this input causes all outputs (regardless of line address) to be blanks (zeros) for characters for which CA6 and CA5 are both 0. A low level input selects normal operation. Inoperative in the graphics mode.
D9 – D0	15 – 24	O	**Data Outputs:** Provide the data for the specified character and line.
V_{CC}	28	I	+ 5V power supply.
GND	14	I	Ground.

Seven bits of an 8-bit address code are used to select 1 of the 128 available characters. The eighth bit functions as a chip enable signal. Each character is defined by a pattern of logic 1's and 0's stored in a 10 x 9 matrix. When a specific 4-bit binary line address code is applied, a word of 10 parallel bits appears at the output. The lines can be sequentially selected, providing a 9-word sequence of 10 parallel bits per word for each character selected by the address inputs. As the line address inputs are sequentially addressed, the device will automatically place the 10 x 9 character in 1 of 2 pre-programmed positions on the 16-line matrix with the positions defined by the 4-line address inputs. One or more of the 10 parallel outputs can be used as control signals to selectively enable functions such as half-dot shift, color selection, etc.

The SCN2670 DCGG includes latches to store the character address and line address data. A control input to inhibit character data output for certain groups of characters is also provided. The SCN2670 also includes a graphics capability, wherein the 8-bit character code is translated directly into 256 possible user programmable graphic patterns. Thus, the DCGG can generate data for 384 distinct patterns, of which 128 are defined by the mask programmable ROM. See figure 1 for a typical applications display.

FUNCTIONAL DESCRIPTION

The DCGG consists of nine major sections. Line and character codes are strobed into the line and character latches. The character latch outputs are presented to the three sources of data; the ROM through an address decoder, the graphics logic, and the output inhibit control. The output inhibit control (together with the SCD input) suppresses the ROM data for selected character codes. The outputs from the line latch drive the line address translation ROM which maps the character ROM data onto 9 of 16 line positions. Finally, the line select multiplexers route the ROM or graphics data to the output drivers on D0 through D9.

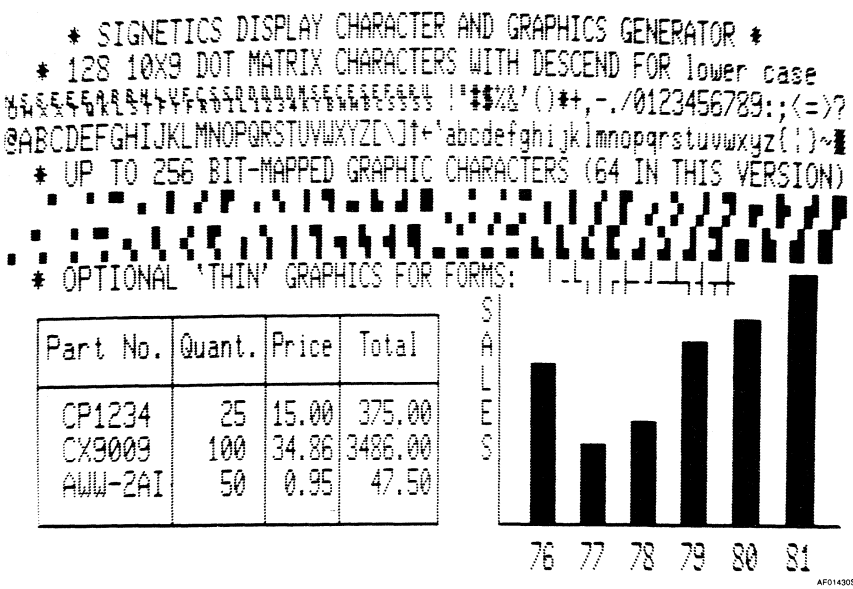


Figure 1. Typical Application Using The 7 x 9 Character Font

Character Latch

The character latch is a 9-bit edge triggered latch used to store the character address (CA0 through CA7) and graphics mode (GM) inputs. The data is stored on the falling edge of CSTROBE. Seven latched addresses (CA0 through CA6) are inputs to the ROM character address decoder. In character mode (GM = 0), CA7 operates as a chip enable. The output drivers are enabled when CA7 = 1 and are three-stated when CA7 = 0. In graphics mode (GM = 1), the output drivers are always enabled and the CA0 through CA7 outputs of the latch are used to generate graphic symbols.

Character Address Decoder

This circuit decodes the 7-bit character address from the character latch to select one of the 128 character fonts stored in the ROM section of the DCGG.

Read Only Memory

The 11,648-bit ROM stores the fonts for the 128 matrix-defined characters. The data for each character consists of 91 bits. Ninety bits represent the 10X9 matrix and one bit specifies whether the character data is output at the normal (unshifted) lines or at the descended (shifted) lines. The 90 data bit outputs are supplied to the line select multiplexers. The descend control bit is an input to the line address translation ROM.

Graphics Logic

When the GM input is zero (low), the DCGG operates in the character mode. When it is one (high), it operates in the graphics mode. In graphics mode, output data is generated by the graphics logic instead of the ROM. The graphics logic maps the latched character address (CA0 through CA7) to the outputs (D0 through D9) as a function of line address (LA0 through LA3). For any particular line address value, two of the CA bits are output: CA0, CA2, CA4 or CA6 is output on D0 through D4 and CA1, CA3, CA5 or CA7 is output on D5 through D9. The outputs are paired: When CA0 is output on D0 through D4, CA1 is output on D5 through D9 and likewise for CA2 – CA3, CA4 – CA5 and CA6 – CA7.

A ROM within the graphics logic allows the specific line numbers for which each pair of bits is output to be specified by the customer. Figure 2 illustrates the general format for graphics symbols and an example where CA7 through CA0 = H'65'. The outputs from the graphics logic go to the line select multiplexers. The multiplexers route the graphic symbol data to the outputs when GM = 1.

Thin Graphics Option

As a customer specified option, 16 of the possible graphic codes (H'80' to H'8F') may be used to generate the special graphic characters illustrated in figure 3. For each of these characters, the vertical component appears on the D4 output. The horizontal component occurs on L_H which is specified by the customer. The vertical components specified by CA0 and CA2 are output for line addresses zero through L_H and L_H through fifteen, respectively.

Line Select Multiplexers

The ten line select multiplexers select ROM data as specified by the line address translation ROM when GM = 0, or graphics data when GM = 1. The inputs to each multiplexer are the nine line outputs from the ROM, an output from the graphics logic and a logic zero (ground).

Output Drivers

Ten output drivers with 3-state capability serve as buffers between the line select multiplexers and external logic. The 3-state control input to these drivers is supplied from the CA7 latch when GM = 0. When GM = 1, the outputs are always active.

Output Inhibit Control

The output inhibit control logic operates only if GM = 0. It causes the output of the line select multiplexers to be logic zero if the SCD input is high and CA6 and CA5 of the latched character address are 00. If the SCD input is low, normal operation occurs. (This feature is useful in ASCII coded applications to selec-

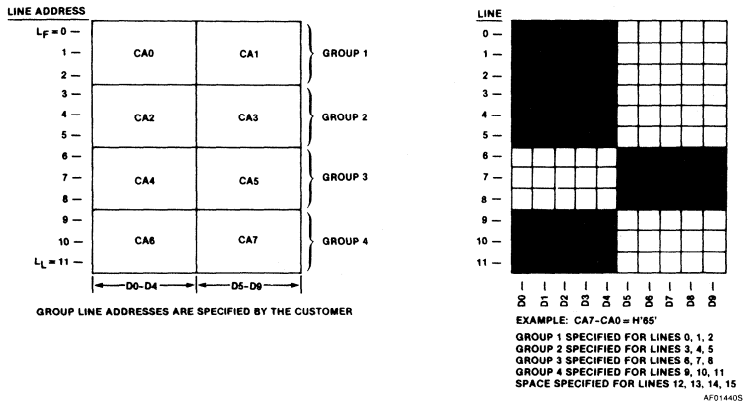


Figure 2. Graphics Symbols — General Format

Table 1. MEMORY MAP FORMAT

2732 Byte Address	Data
0	0 = No thin graphics 1 = Thin graphics
1	Line address for the horizontal segment of the thin line graphics fonts (set to '0' if byte 0 = '0').
2 – 17 (2 – 11H)	Specify group number (1 – 4) to be output at the corresponding line address (0 – F). A '0' specifies no data for that address.
18(12H)	Line address at which the first NONSHIFTED font data appears.
19(13H)	second
20(14H)	third
21(15H)	fourth
22(16H)	fifth
23(17H)	sixth
24(18H)	seventh
25(19H)	eighth
26(1AH)	ninth
27(1BH)	Line address at which the first SHIFTED font data appears.
28(1CH)	second
29(1DH)	third
30(1EH)	fourth
31(1FH)	fifth
32(20H)	sixth
33(21H)	seventh
34(22H)	eighth
35(23H)	ninth
36(24H)	Font truth table printout format 0 = Left to right printing of horizontal data bits D0 – D9 1 = Left to right printing of horizontal data bits D9 – D0
37(25H)	Font truth table printout format 0 = Top to bottom printing of vertical line addresses (0 – FH) 1 = Top to bottom printing of vertical line addresses (F – 0H)
38 – 57 (26 – 39H)	Character 0 data
58 – 77 (3A – 4DH)	Character 1 data
.	.
.	.
.	.
2578 – 2597 (A12 – A25H)	Character 128 (7F H) data

tively disable character generation for non-displayable characters such as line feed, carriage return, etc.)

Line Address Latch

The line address latch is a 4-bit latch used to store the line address (LA0 – LA3). The data is stored on the negative edge of the LSTROBE input.

Line Address Translation ROM

This 32 x 10 ROM translates the 5-bit code consisting of the 4 outputs from the line address latch and the descend control bit from the ROM into a 1-of-10 code for the line select multiplexers. Programming information provided by the customer specifies the address which selects each line of ROM data for both shifted and non-shifted characters. Thus, there are nine line addresses which select ROM data for unshifted characters and nine addresses for shifted characters. These combinations are usually specified by the customer in either ascending or descending order. For the remaining 14 codes (7 each for unshifted and shifted characters), the translation ROM forces zeros at the outputs of the line select multiplexers.

This circuitry only operates if GM = 0. When GM = 1, the line select multiplexers are forced to select the outputs from the graphics logic.

Figure 4 shows an example of data outputs where the customer has specified line 1 as the first line for unshifted characters, line 4 as the first line for shifted characters and line address combinations in ascending order.

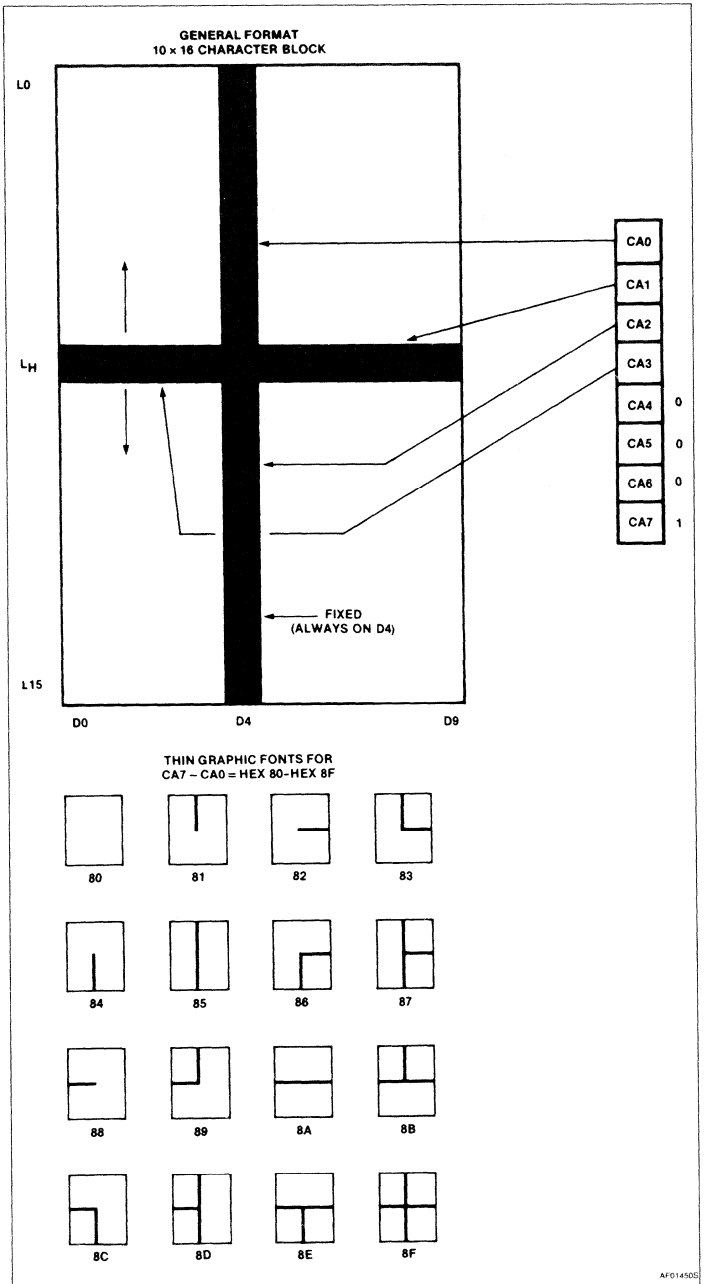


Figure 3. Special Graphic Characters

5 x 7 Design Note

The 5 x 7 character fonts of the 2670C have been centered in the 10 x 9 character fields. Because of this centering, only the D1 through D8 data bus pins (pins 23 through 16) are to be used for the character data. D1 can be connected to the least significant bit of the Signetics Video Attributes Controller data bus (D0), D2 of the 2670C can be connected to D1 of the attributes controller, etc. The D0 and D9 pins will always be read as zero and will not be connected.

CUSTOM PATTERN PROGRAMMING INSTRUCTIONS

Signetics requires that the customer supply them with a 2732 EPROM containing the necessary data for specifying a custom version of the SCN2670. The 2732 will contain the data to be stored in the ROM array, the programmable line address translation ROM, thin graphics option and the graphics line font translation ROM. Signetics will no longer accept card decks.

The memory map format for programming the 2732 is contained in table 1.

Bytes 0 and 1 relate to the thin line graphics option. Thin line graphics consists of two components: a vertical and a horizontal segment. The vertical segment always occurs at the D4 output. The location of the horizontal segment (line addresses 0 – 15) is specified by the customer in byte 1.

Bytes 2 through 17 correspond to how the graphics blocks are created from the 16 horizontal addresses. A '0' specifies no data for that address. Figure 5 shows an example of how the character field can be divided into the graphics blocks. The line addresses can be divided into a maximum of four groups.

Since line addresses A through F are not used, byte addresses 12 through 17 are set to '0'. Line addresses 0 and 1 are part of group 1, so byte addresses 2 and 3 are set equal to '1'. Line addresses 2 through 4 are in group 2, so byte addresses 4 through 6 are set equal to '2'. The same procedure is followed for the remaining line addresses. The group numbering assignments must be sequential (i.e., line address 1 cannot be in group 2 if line addresses 0 and 2 are in group 1).

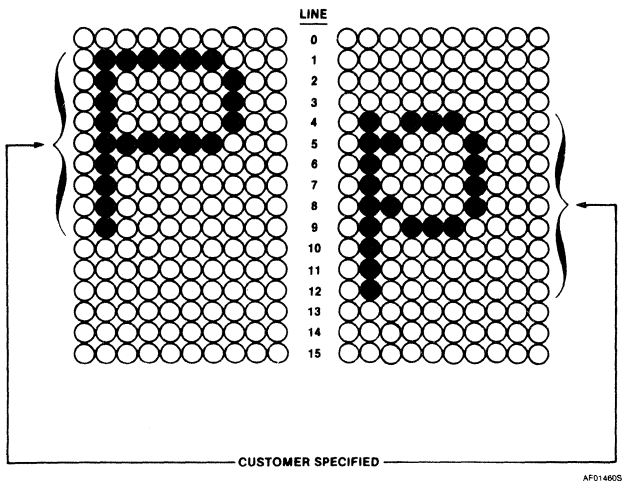


Figure 4. Customer Specified Example

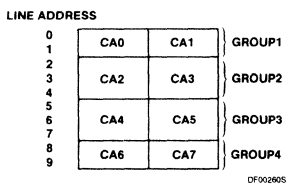


Figure 5. Character Field Divisions

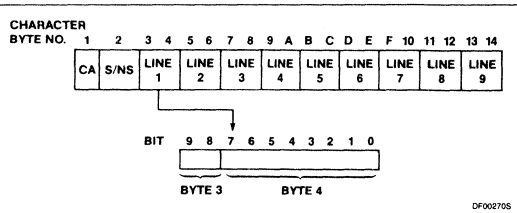


Figure 6. Data Format

Byte Addresses 18 through 26 are used to indicate which line address the nonshifted character fonts begin and end. The first line of the character font must occur from line address 0 to 7. If the character font begins at line address 2, the byte addresses are set up as follows:

BYTE ADDRESS	DATA
18	2
19	3
20	4
21	5
22	6
23	7
24	8
25	9
26	10

Byte Addresses 27 through 35 are used to indicate which line address the shifted character fonts begin and end. Again the first line of the character font must occur from line addresses 0 through 7. The data is entered the same as for the nonshifted character fonts above.

Bytes 36 and 37 are only used for printouts of the ROM codes by Signetics.

The remaining byte addresses (38 – 2597) contain the data for the 128 characters. Each character consists of 20 bytes. The format is contained in figure 6.

Character Byte 1 (CA) contains the character address (CA6 – CA0).

For character byte 2, if S/NS = '1', the character font will begin at the shifted location as specified by byte addresses 27 – 35. If S/NS = '0', then the character font will begin at the nonshifted location as specified by byte addresses 18 – 26.

The remaining character bytes define the nine line character font. The bytes are paired as each font line requires 10 bits of information to be output on lines D9 – D0 (see figure 6). Remember, when coding each line, D9 is the rightmost bit of the character line, and D0 is the leftmost bit.

Printouts

Signetics will submit the following printouts to the customer for approval:

- A repeat of all customer information.
- A separate font drawing for each of the 128 ROM characters and 256 graphics fonts. The font drawings are positioned on a 10 x 16 matrix as specified by the customer's translation data.

Customer Example

A question mark has a character address of 3F hex. The character code for it is as follows:

```
L0    . . X X X X X . . .
L1    . X . . . . . X . .
L2    . X . . . . . X . .
L3    . . . . . . . X . .
L4    . . . . . X X . . .
L5    . . . . X . . . . .
L6    . . . . X . . . . .
L7    . . . . . . . . . .
L8    . . . . X . . . . .
bit   0 1 2 3 4 5 6 7 8 9
```

Character

Byte 1:	3F			
2:	00			(nonshifted)
3:	00	4:	C7	(Line 1)
5:	00	6:	28	(Line 2)
7:	00	8:	28	(Line 3)
9:	00	A:	08	(Line 4)
B:	00	C:	06	(Line 5)
D:	00	E:	01	(Line 6)
F:	00	10:	01	(Line 7)
11:	00	12:	00	(Line 8)
13:	00	14:	01	(Line 9)

We recommend that the character fonts be created with respect to the center of the 10 x 16 character field (data bit D4) for optimum symmetry, especially if thin line graphics is implemented as the vertical segment is always located at bit D4.

ABSOLUTE MAXIMUM RATINGS [1]

PARAMETER	RATING	UNIT
Supply voltage	6.0	V
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.3 to +6.0	V

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on + 150°C maximum junction temperature and thermal resistance of 60°C/W junction to ambient (ceramic package).
3. This product includes circuitry specifically designed for the protection of its internal devices from the damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the maxima.

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5.0V \pm 5\%$ [1,2,3]

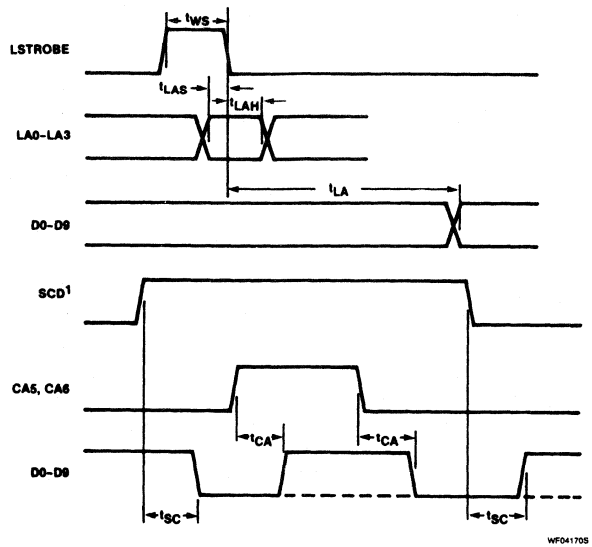
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_O = 1.6mA$	0		0.4	V
V_{OH}	Output high voltage	$I_O = -100\mu A$	2.4		V_{CC}	V
I_{IL}	Input leakage current	$V_{IN} = 0$ to 4.25V			10	μA
I_{OL}	Output leakage current	$V_O = 0.4$ to 4V	−10		+10	μA
I_{CC}	Supply current	$V_{CC} = 5.25V$		35	80	mA
C_{IN}	Input capacitance	All other pins grounded			10	pF
C_{OUT}	Output capacitance				15	pF

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5.0V \pm 5\%$ [1,2,3,4]

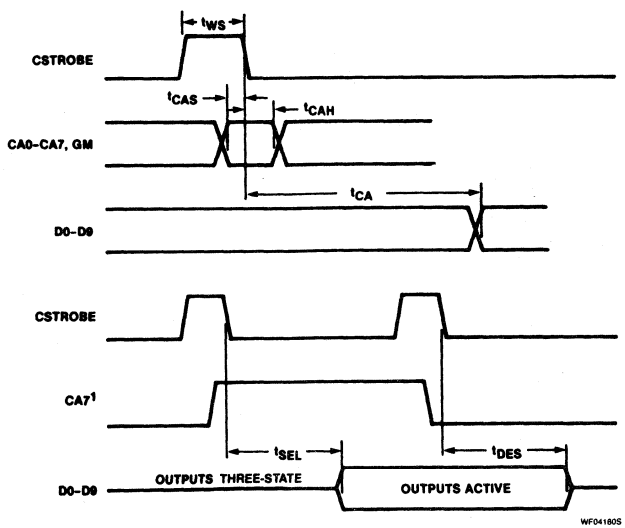
PARAMETER		LIMITS				UNIT
		300ns		200ns		
		Min	Max	Min	Max	
t_{WS}	Strobe pulse width	100		100		ns
t_{LAS}	Line address set-up	50		50		ns
t_{LAH}	Line address hold	25		25		ns
t_{CAS}	Character address set-up	25		15		ns
t_{CAH}	Character address hold	25		15		ns
t_{CA}	Character select access	30	300	30	200	ns
t_{LA}	Line select access	30	500	30	350	ns
t_{SEL}	Chip select delay		250		150	ns
t_{DES}	Chip deselect delay		200		125	ns
t_{SC}	Special character blank/unblank time		300		200	ns

NOTES:
1. Parameters are valid over operating temperature range unless otherwise specified.
2. All voltage measurements are referenced to ground. All time measurements are at the 0.8V or 2.0V level for inputs and outputs. Input levels are 0V and 2.4V.
3. Typical values are at + 25°C, typical supply voltages and typical processing parameters.
4. Test conditions: $C_L = 100pF$ and 1 TTL load.

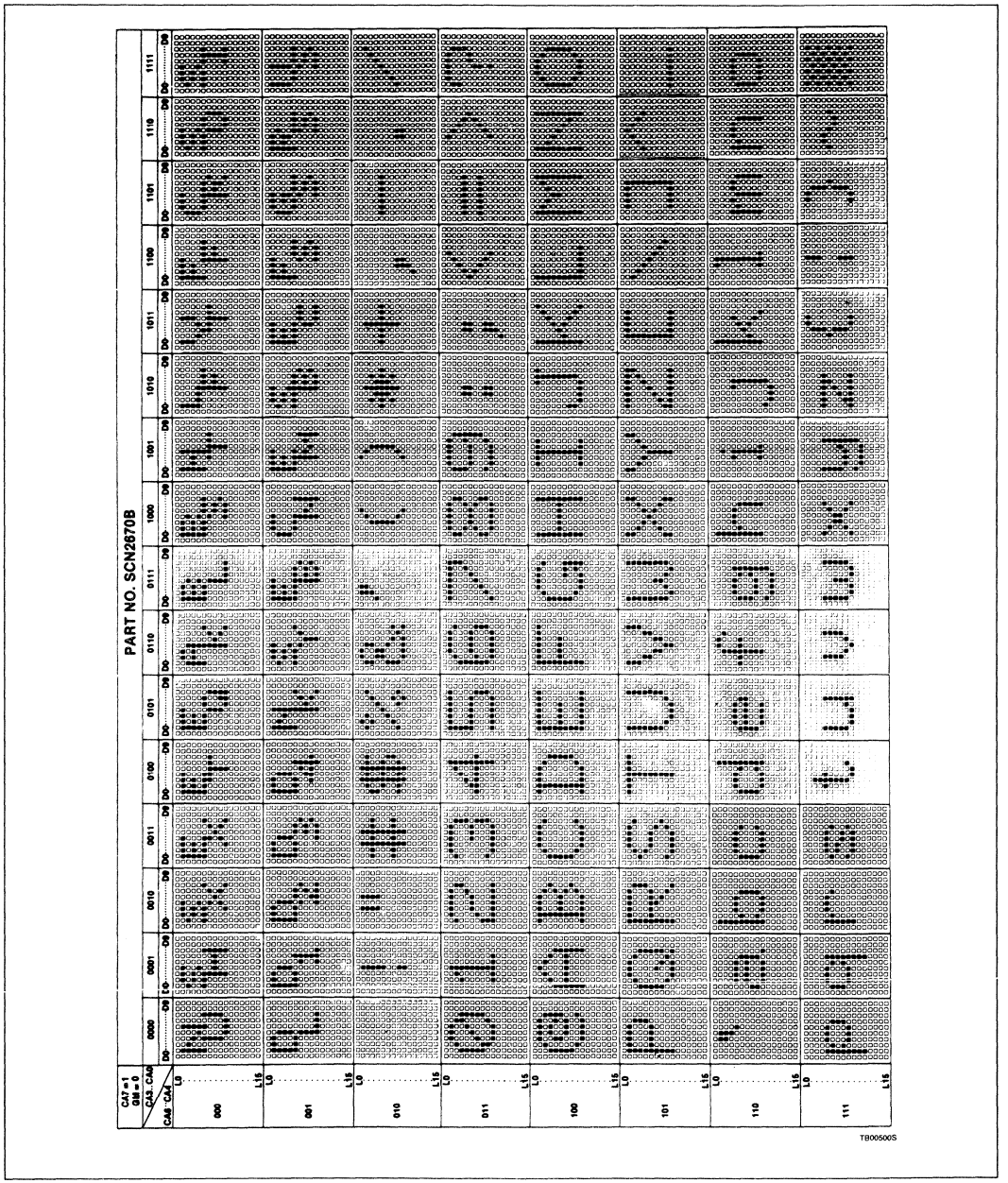
TIMING DIAGRAMS

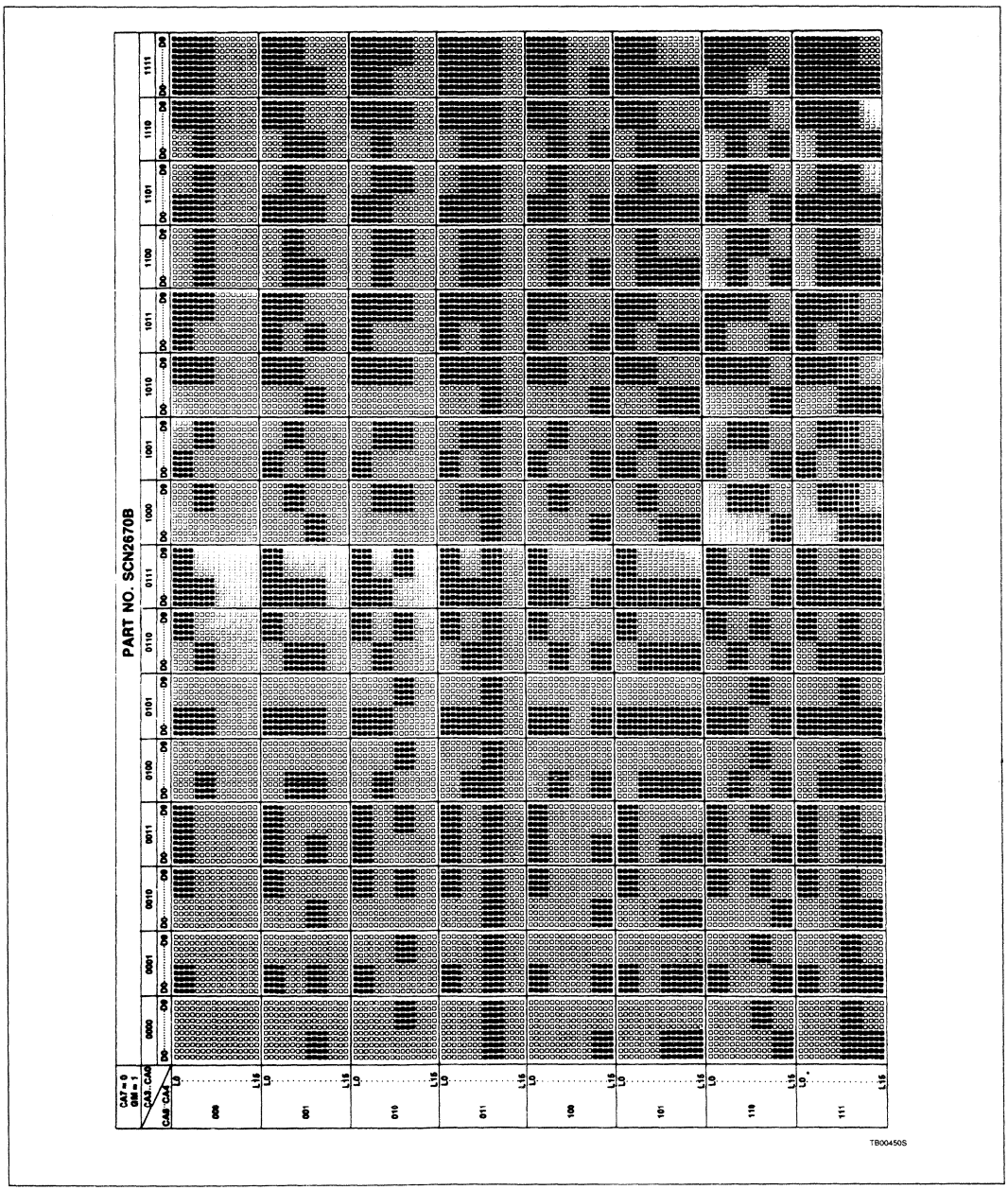


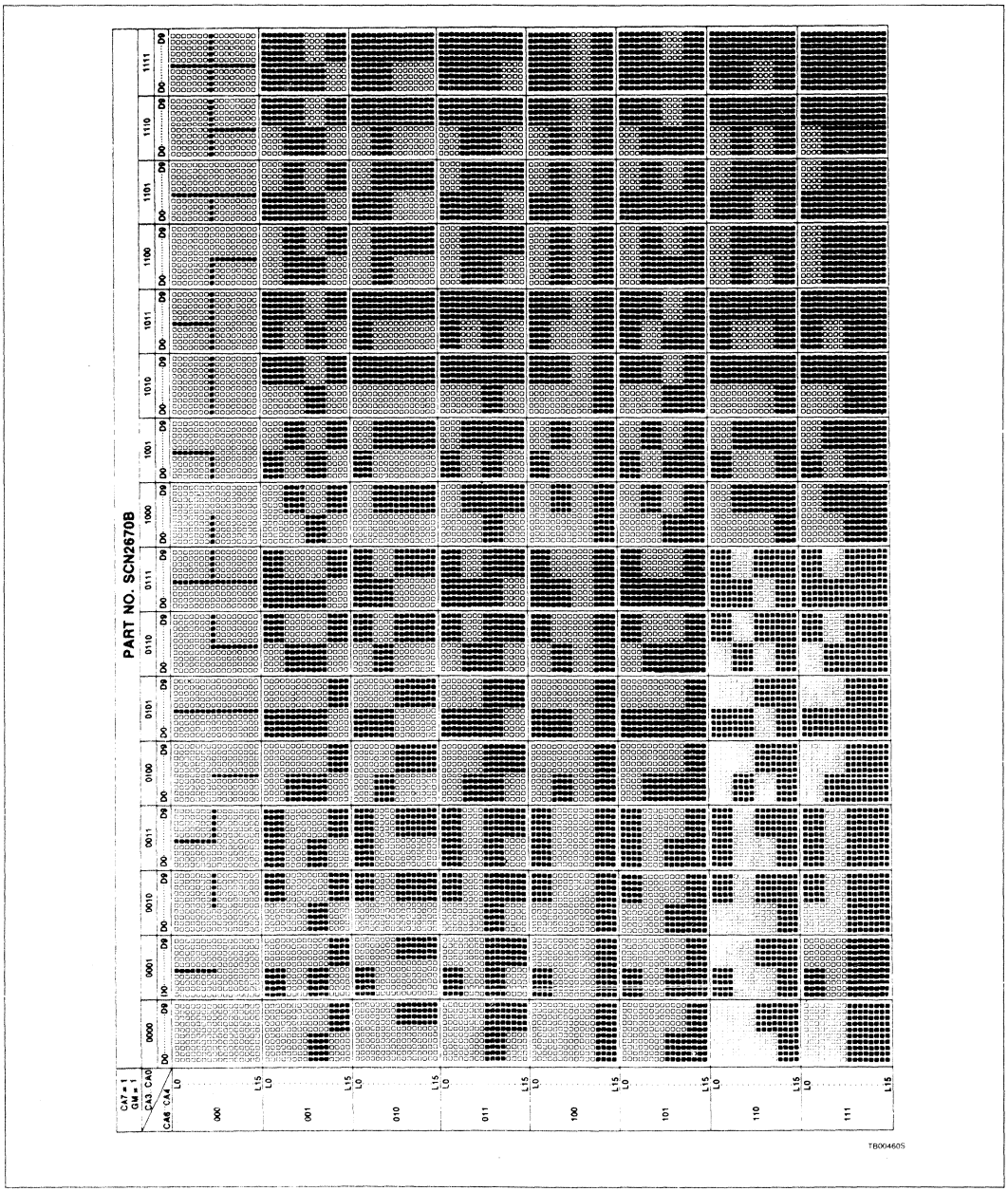
NOTE:
1. When GM = 1, SCD input is inactive.

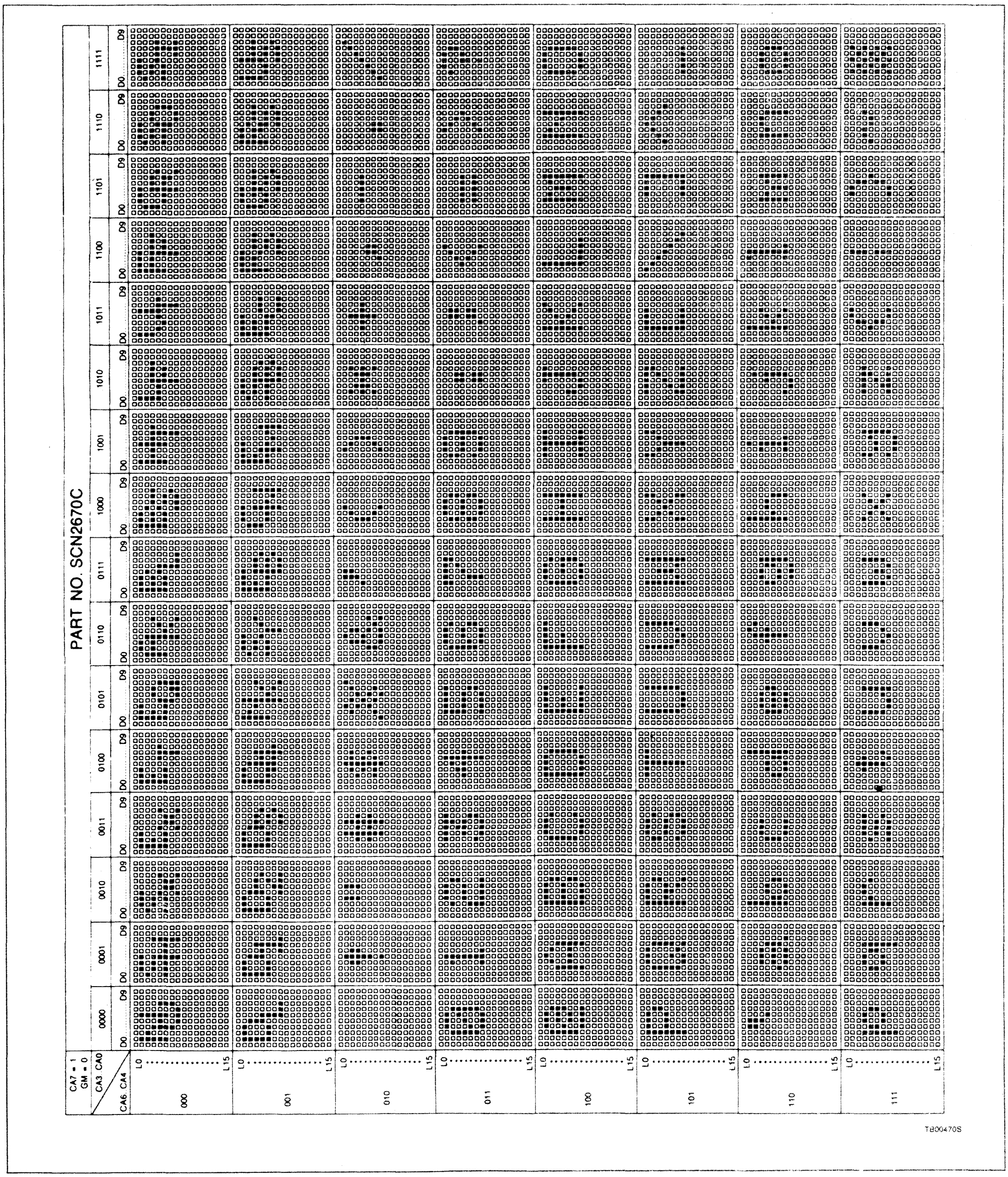


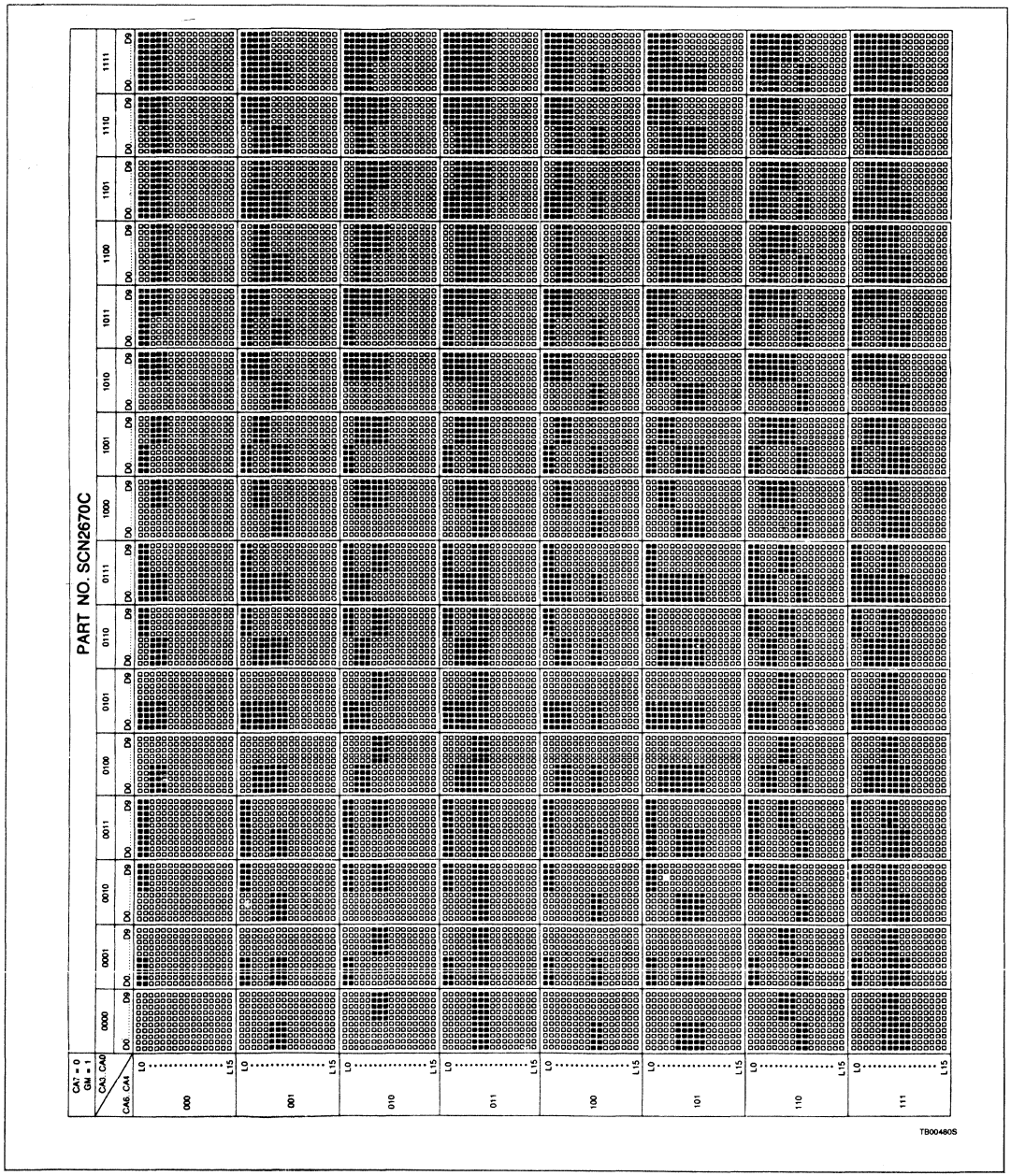
NOTE:
1. CA7 operates as output enable only in character mode (GM = 0).

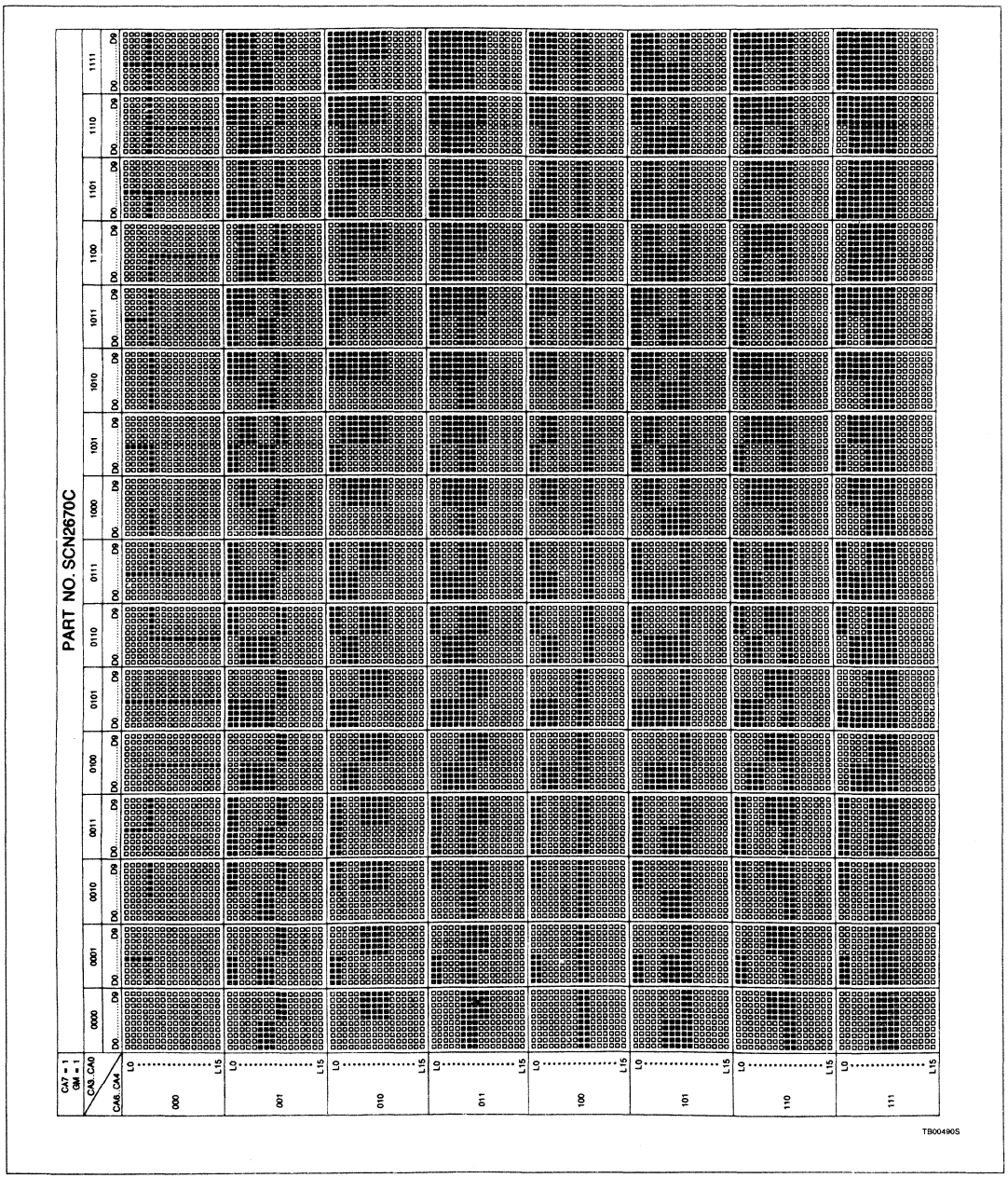

PART NO. SCN2670B
CA7 = 1
QM = 0
CA3 CA0
CA6 CA4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
D0
D7
L0
L15
000
001
010
011
100
101
110
111
TB00500S


PART NO. SCN2670B
CAT = 0
GM = 1
CA3 CA2 CA1 CA0
000
001
010
011
100
101
110
111
L0
L15
D0
D6
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
TB00450S


PART NO. SCN2670B
CA7 = 1
CA7 = 0
CA3 CA0
CA6 CA4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
D0
D9
L0
L15
000
001
010
011
100
101
110
111
TB00460S


PART NO. SCN2670C
CA7 = 1
GM = 0
CA3 CA0
CA6 CA4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
D0
D9
L0
L15
000
001
010
011
100
101
110
111
TB00470S


PART NO. SCN2670C
CAT = 0
CM = 1
CA3 CA0
CA6 CA4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
D0
D9
L0
L15
000
001
010
011
100
101
110
111
TB00480S


PART NO. SCN2670C
CA7 = 1
GM = 1
CA3..CA0
CA6..CA4
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
D0
D9
L0
L15
000
001
010
011
100
101
110
111
TB00490S

Programmable Keyboard and Communications Controller (PKCC)

Product Specification

Originally published by Signetics February 1985

DESCRIPTION

The Signetics SCN2671 Programmable Keyboard and Communications Controller (PKCC) is an MOS LSI device which provides a versatile keyboard encoder and an independent full duplex asynchronous communications controller. It is intended for use in microprocessor based systems and provides an eight bit data bus interface.

The keyboard encoder handles the scanning, debounce, and encoding of a keyboard matrix with a maximum of 128 keys. It provides four levels of key encoding corresponding to the separate SHIFT and CONTROL input combinations. Four keyboard rollover modes can be programmed including provisions for up to 16 latched keys. Control outputs are provided for interfacing with contact or capacitive keyboards. An eight bit keyboard status register provides status information to the CPU.

The receiver section of the communications controller accepts serial data from the RxD pin and converts it to parallel data characters. Simultaneously, the transmitter section accepts parallel data from the data bus and outputs serialized data onto the TxD pin. Received data is checked for parity and framing errors, and break conditions are flagged. Character lengths can be programmed as 5, 6, 7, or 8 bits not including parity, start or stop bits. An internal baud rate generator (BRG) with 16 divider ratios can be used to derive the receive and/or transmit clocks. The BRG can accept an external clock or operate directly from a crystal. An eight bit communications status register provides status information to the CPU.

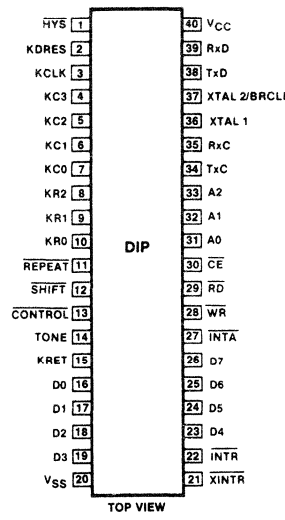
FEATURES

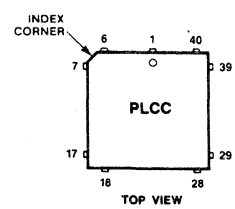
- **Keyboard Interface**
  - **Contact or capacitive keyboard**
  - **Up to 128 keys on an 8 x 16 matrix**
  - **Encoded or unencoded operation**
  - **Four code levels per key**
  - **Latched key option–separate depress and release codes**
  - **Programmable scan rate and debounce time**
  - **Programmable rollover modes**
  - **Programmable auto-repeat for selected keys**
  - **Tone output–two frequencies**
- **Asynchronous communication interface**
  - **Internal baud rate generator–16 rates**
  - **Full duplex operation**
  - **Detection of start and end of break**
  - **Programmable break generation**
  - **Programmable character parameters**
  - **Auto-echo and maintenance loopback modes**
- **Polled or interrupt operation**
- **Interrupt priority controller and vector generator**
- **Operates directly from crystal or external clocks**
- **TTL compatible**
- **Single + 5 volt power supply**

APPLICATIONS

- **CRT terminals**
- **Hard copy terminals**
- **Word processing systems**
- **Data entry terminals**
- **Small business computers**

PIN CONFIGURATION



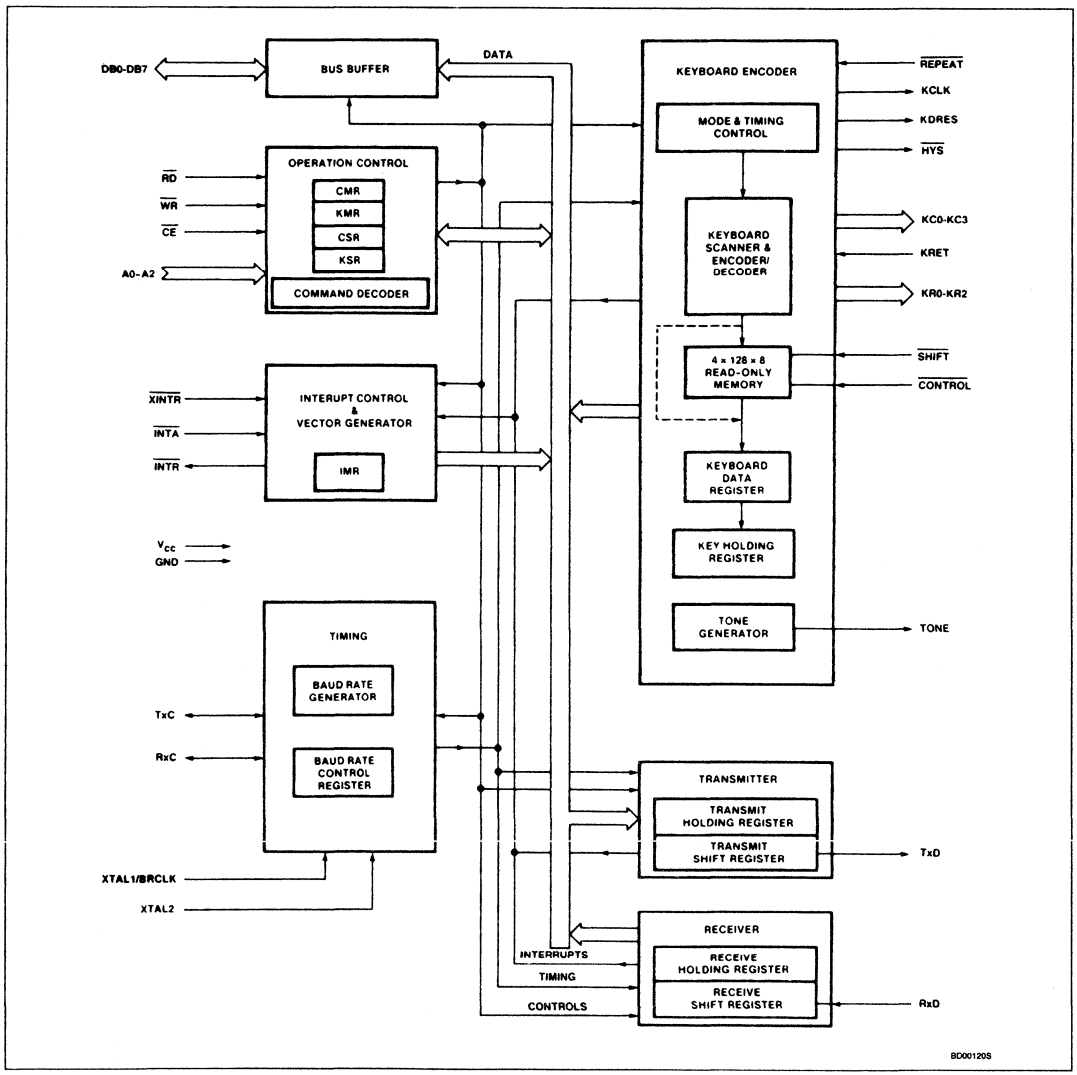


Pin	Function	Pin	Function
1	NC	23	XINTR
2	HYS	24	INTR
3	KDRES	25	D4
4	KCLK	26	D5
5	KC3	27	D6
6	KC2	28	D7
7	KC1	29	INTA
8	KC0	30	WR
9	KR2	31	RD
10	KR1	32	CE
11	KR0	33	A0
12	NC	34	NC
13	REPEAT	35	A1
14	SHIFT	36	A2
15	CONTROL	37	TxC
16	TONE	38	RxC
17	KRET	39	XTAL1
18	D0	40	XTAL2/BRCLK
19	D1	41	TxD
20	D2	42	RxD
21	D3	43	NC
22	VSS	44	VCC

ORDERING CODE

PACKAGES	V_{CC} = 5V±5%, T_A = 0°C to 70°C
Ceramic DIP	SCN2671AC1I40
Plastic DIP	SCN2671AC1N40
Plastic LCC	SCN2671AC1A44

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.	TYPE	NAME AND FUNCTION
D0 – D7	16 – 19, 23 – 26	I/O	**Data Bus:** 8-bit three-state bidirectional data bus. All data, command and status transfers are made using this bus. D0 is the least significant bit; D7 is the most significant bit.
A0 – A2	31 – 33	I	**Address Lines:** Used to select internal PKCC registers or commands.
$\overline{RD}$	29	I	**Read Strobe:** When low, gates the selected PKCC register onto the data bus if $\overline{CE}$ is also low.
$\overline{WR}$	28	I	**Write Strobe:** When low, gates the contents of the data bus into the selected PKCC register if $\overline{CE}$ is also low.
$\overline{CE}$	30	I	**Chip Enable:** When high, places the D0 – D7 output drivers in a three-state condition. If $\overline{CE}$ is low, data transfers are enabled in conjunction with the $\overline{RD}$ and $\overline{WR}$ inputs.
$\overline{INTR}$	22	O	**Interrupt Request:** Several conditions may be programmed to request an interrupt to the CPU. It is an active low open-drain output. This pin will be inactive after power on reset or a master reset command.
$\overline{INTA}$	27	I	**Interrupt Acknowledge:** Used to indicate that an interrupt request has been accepted by the CPU. When $\overline{INTA}$ goes low, the PKCC outputs an 8-bit address vector on D0 – D7 corresponding to the highest priority interrupt currently active.
$\overline{XINTR}$	21	I	**External Interrupt:** An active low external interrupt input to the PKCC interrupt priority resolver.
TxC	34	I/O	**Transmitter Clock:** The function of this pin depends on bit 7 of the baud rate control register (BRR7). If external transmitter clock is selected (BRR7 = 0), it is an input for the transmitter clock. If internal transmitter clock is selected (BRR7 = 1), this pin is an output which is a multiple of the actual baud rate (1X, 16X) as selected by BRR5. The data is transmitted on the falling edge of TxC. It is an input after power on and after master reset or communications reset commands.
RxC	35	I/O	**Receiver Clock:** The function of this pin depends on BRR6. If external receiver clock is selected (BRR6 = 0), it is an input for the receiver clock. If internal receiver clock is selected (BRR6 = 1), this pin is an output which is a multiple of the actual baud rate (1X, 16X) as selected by BRR4. The received data is sampled on the rising edge of RxC. It is an input after power on and after master reset or communications reset commands.
TxD	38	O	**Transmitter Data:** This output is the transmitted serial data; the least significant bit is transmitted first. This pin is high after power on reset or a reset command that affects the transmitter.
RxD	39	I	**Receiver Data:** This input is the serial data input to the receiver. The least significant bit is received first.
XTAL1, XTAL2/BRCLK	36,37	I	**Connections for Crystal:** Provides an on-chip clock generator for the internal baud rate generator and the keyboard interface logic. If an external clock is provided, use XTAL2 as the clock input. See figures 20 and 21. All timing parameters such as keyboard scan time, tone frequency, and baud rate assume a clock input at the specified BRG input frequency. If this frequency is different the timing parameters will vary proportionately.
KR0 – KR2	10 – 8	O	**Keyboard Row Scan:** Decoded externally; selects one of eight rows.
KC0 – KC3	7 – 4	O	**Keyboard Column Scan:** Decoded externally; selects one of 16 columns.
KRET	15	I	**Key Return:** An active high level indicates that the key being scanned is closed.
$\overline{SHIFT}$	12	I	**SHIFT Key:** Active low input from the SHIFT key. The combination of $\overline{SHIFT}$ and $\overline{CONTROL}$ inputs select one of four possible codes from the internal key encoding ROM.
$\overline{CONTROL}$	13	I	**CONTROL Key:** Active low input from the CONTROL key. The combination of $\overline{SHIFT}$ and $\overline{CONTROL}$ inputs select one of four possible codes from the internal key encoding ROM.
$\overline{REPEAT}$	11	I	**REPEAT Key:** Active low input from the $\overline{REPEAT}$ key. Causes the key depression currently active to be repeated at a rate of approximately 15 times per second.
KCLK	3	O	**Keyboard Clock:** High frequency (approximately 400kHz) output used to scan capacitive keyboards.
KDRES	2	O	**Key Detect Reset:** Resets the analog detector before scanning a key. Used for capacitive keyboards.
$\overline{HYS}$	1	O	**Hysteresis Output:** Sent to the analog detector for capacitive keyboard applications. A low indicates the key currently being scanned has been recognized on previous scan cycles.
TONE	14	O	**Square Wave Output:** Used for tone generation.
V_{CC}	40	I	+5V power supply.
V_{SS}	20	I	Ground.

The PKCC has an interrupt mask register to selectively enable certain keyboard and communications status bits to generate interrupts. Priority encoded interrupt vectoring is available. Upon receipt of an interrupt acknowledge, an interrupt vector will be output on D0 – D7 reflecting the source of the interrupt. The interrupt source can also be read from an interrupt status register.

FUNCTIONAL DESCRIPTION

The PKCC consists of six major sections (see block diagram). These are the transmitter, receiver, timing, operation control, keyboard encoder, and a priority encoded interrupt control unit. These sections communicate with each other via an internal data bus and an internal control bus. The internal data bus interfaces to the microprocessor data bus via a bidirectional data bus buffer.

Operation Control

This functional block stores configuration and operation commands from the CPU and generates appropriate signals to various internal sections to control the overall device operation. It contains read and write circuits to permit communications with the microprocessor via the data bus and contains mode registers KMR and CMR, the command decoder, and status registers KSR and CSR. Details of operating modes and status information are presented in the Operation section of this data sheet. The register addressing is specified in table 1.

Timing

The PKCC contains a baud rate generator (BRG) which is programmable to accept external transmit or receive clocks or to divide an external clock to perform data communications. The unit can generate 16 baud rates, any of which can be selected for full duplex operation. The external clock to the baud rate generator can be applied directly to the XTAL2 input (see figure 21) or can be generated internally by connecting a crystal across the XTAL1, XTAL2 input pins. The clock input is also utilized by the keyboard encoder section. Thus, a clock must be provided even if external transmitter and receiver clocks are used.

Receiver

The receiver accepts serial data on the RxD pin, converts this serial input to parallel format, checks for break conditions, framing and parity errors, and loads an "assembled" character in the receive holding register for access by the CPU.

Transmitter

The transmitter accepts parallel data loaded by the CPU into the transmit holding register and converts it to a serial bit stream framed by the start bit, calculated parity bit (if specified), and stop bit(s). The composite serial stream of data is transmitted on the TxD output pin.

Table 1. REGISTER ADDRESSING

$\overline{CE}$	A2	A1	A0	$\overline{RD}/\overline{WR}$	FUNCTION
1	X	X	X	X	Three-state data bus
0	0	0	0	$\overline{WR}$	Reset command (see table 6)
0	0	0	0	$\overline{RD}$	Read interrupt status register (ISR)
0	0	0	1	$\overline{RD}$, $\overline{WR}$	Read/write communications mode register (CMR)
0	0	1	0	$\overline{WR}$	Write transmit holding register (TxHR)
0	0	1	0	$\overline{RD}$	Read receiver holding register (RxHR)
0	0	1	1	$\overline{WR}$	Write baud rate mode register (BRR)
0	0	1	1	$\overline{RD}$	Read communications status register (CSR)
0	1	0	0	$\overline{RD}$, $\overline{WR}$	Read/write interrupt mask register (IMR)
0	1	0	1	$\overline{RD}$, $\overline{WR}$	Read/write keyboard mode register (KMR)
0	1	1	0	$\overline{RD}$	Read keyboard holding register (KHR)
0	1	1	1	$\overline{RD}$	Read keyboard status register (KSR)
0	1	1	1	$\overline{WR}$	Miscellaneous commands (see description)

NOTE:
X = don't care.

Keyboard Encoder

The keyboard encoder provides encoded scanning signals for a matrix keyboard. Key depressions are detected on the KRET input. The debounced and verified key codes (or matrix addresses) are loaded into the key holding register for access by the CPU. Figures 1 and 2 illustrate the PKCC interface to contact and capacitive keyboards, respectively.

Interrupt Control

The interrupt controller unit contains a software programmable interrupt mask register which selectively enables status conditions from the keyboard encoder and communication controller to generate interrupts. The interrupts are priority encoded and individually generate an eight bit vector which is output on the data bus in response to a CPU interrupt acknowledge on the $\overline{INTA}$ input pin.

OPERATION

Keyboard Encoder

The keyboard is continuously scanned by KC0 – KC3 and KR0 – KR2 which are decoded externally to handle 128 possible keys (see figures 1 and 2). KC0 – KC3 select one of 16 columns and KR0 – KR2 multiplex the eight row return lines into the KRET pin. Debouncing is accomplished by remembering a 1 state at the KRET pin when a key is being addressed and verifying it one scan later. Once the key is verified, a key code is loaded into the keyboard data register (KDR). If the keyboard holding register (KHR) is empty, the contents of the KDR will be transferred to the KHR immediately; if the KHR is full (i.e., the CPU has not read the previous key code), the transfer will be held off until the KHR is read. The data transfer to the KHR causes keyboard data ready (KRDY) to be set in the keyboard status register.

For capacitive keyboards, the high frequency output KCLK can be used to gate the column scan to the keyboard (see figure 2). The key detector reset (KDRES) output resets the analog detector prior to scanning each key location. The output from the analog multiplexer is sensed and then latched in the analog detector. The $\overline{HYS}$ output controls the sense level. A 0 will lower the sense level causing hysteresis, and a 1 will raise the sense level with no hysteresis.

The $\overline{REPEAT}$ input enables the keyboard logic to recognize any key repeatedly, 15 times per second. Additionally, certain keys can be programmed to repeat automatically if depressed for more than one-half second.

A square wave is output on the TONE pin when the CPU issues a ring tone command to the PKCC.

Keyboard Mode Register

Operating modes are selected by programming the keyboard mode register (KMR), see figure 3. Bit KMR7 is used for testing the device. For normal operation, this bit should always be written to a 0. Bits KMR6 – KMR5 select the rollover modes for keyboard processing:

N-key Rollover: In this mode, the code corresponding to each key depression is loaded into the KDR as soon as that key is debounced, independent of the release of other keys. Two or more closures occurring within one scan cycle are considered to be simultaneous, which will set keyboard error in the keyboard status register (KSR1). As soon as the keyboard holding register is empty, the code in the KDR is transferred to the KHR and the KRDY status bit is set (KSR0).

N-Key Rollover With Latched Keys: This mode is the same as regular N-key rollover,

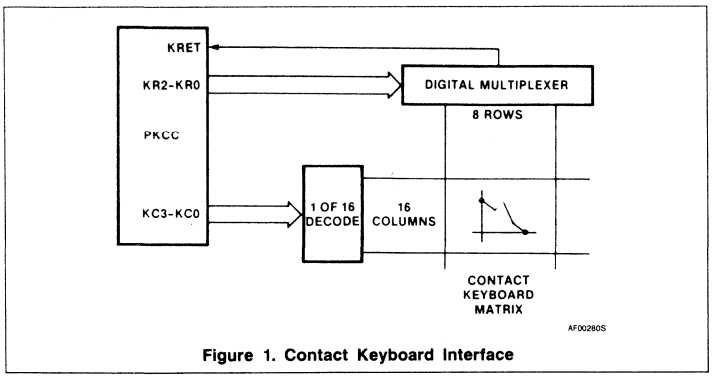


Figure 1. Contact Keyboard Interface

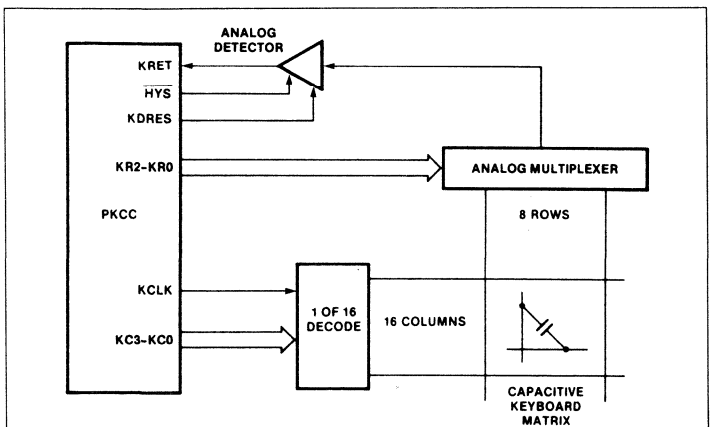


Figure 2. Capacitive Keyboard Interface

except that the keys which are assigned to row 0 of the keyboard matrix (KR2 – KR0 = 000) produce a code both when depressed and when released. The codes are independent of the states of the inputs at $\overline{\text{SHIFT}}$ and $\overline{\text{CONTROL}}$. If one or more of the latched keys are depressed when the keyboard is enabled (after a keyboard reset), the corresponding codes will be sent out as the keys are scanned and debounced. Note that simultaneous latched keys will not set KERR (KSR1) and that latched keys will not be auto-repeat and will not be affected by the $\overline{\text{REPEAT}}$ input.

Two-Key Rollover: The first key code is loaded into the KDR immediately and the second code is loaded only after the first key is released. Simultaneous keys will set KERR (KSR1), if three or more keys remain closed at any given time, the KERR bit will also be set. All keys must then be released before the next KRET will be processed.

Two-Key Inhibit: All keys must be released between keystrokes; otherwise, KERR (KSR1) will be set.

Bit KMR4 specifies the key encoding mode. Each key is assigned four 8-bit codes, corresponding to the states of the $\overline{\text{SHIFT}}$ and $\overline{\text{CONTROL}}$ inputs. If the encoded mode is programmed, the row/column address of the detected key is used to load one of the four key codes into the KDR. See table 2 for key code assignments. If the non-encoded mode is programmed, the row/column address is loaded directly into the KDR with the following format:

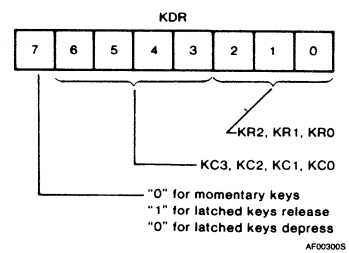
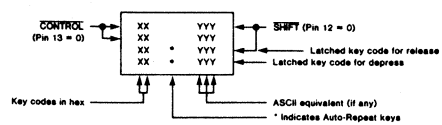


Table 2. STANDARD KEY CODES (HEX)

COLUMN (KC3 – KC0)	ROW (KR2 – KR0)							
	0	1	2	3	4	5	6	7
0	E0	C0	1B ESC	09 HT	1F US	1A SUB	30 0	2B +
	F0	D0	1B ESC	09 HT	1F US	1A SUB	30 0	3B ;
	E0	C0	1B ESC	09 * HT	1F US	5A Z	30 0	2B +
	F0	D0	1B ESC	09 * HT	1F US	7A z	30 0	3B ;
1	E1	C1	21 !	11 DC1	01 SOH	18 CAN	3D =	2A *
	F1	D1	31 1	11 DC1	01 SOH	18 CAN	2D –	3A :
	E1	C1	21 !	51 Q	41 A	58 X	3D * =	2A *
	F1	D1	31 1	71 q	61 a	78 x	2D * –	3A :
2	E2	C2	22 "	17 ETB	13 DC3	03 ETX	1E RS	1F US
	F2	D2	32 2	17 ETB	13 DC3	03 ETX	1E RS	1F US
	E2	C2	22 "	57 W	53 S	43 C	7E ~	7F * DEL
	F2	D2	32 2	77 w	73 s	63 c	5E ↑	5F * —
3	E3	C3	23 #	05 ENQ	04 EOT	16 SYN	1C FS	1B ESC
	F3	D3	33 3	05 ENQ	04 EOT	16 SYN	1C FS	1B ESC
	E3	C3	23 #	45 E	44 D	56 V	7C ¦	7B {
	F3	D3	33 3	65 e	64 d	76 v	5C \	5B [
4	E4	C4	24 $	12 DC2	06 ACK	02 STX	08 BS	1D GS
	F4	D4	34 4	12 DC2	06 ACK	02 STX	08 BS	1D GS
	E4	C4	24 $	52 R	46 F	42 B	08 * BS	7D }
	F4	D4	34 4	72 r	66 f	62 b	08 * BS	5D]
5	E5	C5	25 %	14 DC4	07 BEL	0E SO	10 DLE	08 BS
	F5	D5	35 5	14 DC4	07 BEL	0E SO	10 DLE	08 BS
	E5	C5	25 %	54 T	47 G	4E N	50 P	08 BS
	F5	D5	35 5	74 t	67 g	6E n	70 p	08 * BS
6	E6	C6	26 &	19 EM	08 BS	0D CR	00 NUL	09 HT
	F6	D6	36 6	19 EM	08 BS	0D CR	00 NUL	09 HT
	E6	C6	26 &	59 Y	48 H	4D M	60 `	09 * HT
	F6	D6	36 6	79 y	68 h	6D m	40 @	09 * HT
7	E7	C7	27 '	15 NAK	0A LF	3C <	7F DEL	20 SP
	F7	D7	37 7	15 NAK	0A LF	2C ,	7F DEL	20 SP
	E7	C7	27 '	55 U	4A J	3C <	7F DEL	20 * SP
	F7	D7	37 7	75 u	6A j	2C ,	7F DEL	20 * SP
8	E8	C8	28 (	09 HT	0B VT	3E >	0A LF	0B VT
	F8	D8	38 8	09 HT	0B VT	2E .	0A LF	0B VT
	E8	C8	28 (	49 I	4B K	3E >	0A LF	0B * VT
	F8	D8	38 8	69 i	6B k	2E .	0A LF	0B * VT
9	E9	C9	29)	0F SI	0C FF	3F ?	0D CR	0A LF
	F9	D9	39 9	0F SI	0C FF	2F /	0D CR	0A LF
	E9	C9	29)	4F O	4C L	3F ?	0D CR	0A * LF
	F9	D9	39 9	6F o	6C l	2F /	0D CR	0A * LF
A	EA	CA	37 7	34 4	31 1	30 0	A0	A6
	FA	DA	37 7	34 4	31 1	30 0	B0	B6
	EA	CA	37 7	34 4	31 1	30 0	A0	A6
	FA	DA	37 7	34 4	31 1	30 0	B0	B6
B	EB	CB	38 8	35 5	32 2	2E .	A1	A7
	FB	DB	38 8	35 5	32 2	2E .	B1	B7
	EB	CB	38 8	35 5	32 2	2E .	A1	A7
	FB	DB	38 8	35 5	32 2	2E .	B1	B7
C	EC	CC	39 9	36 6	33 3	BF	A2	A8
	FC	DC	39 9	36 6	33 3	AF	B2	B8
	EC	CC	39 9	36 6	33 3	9F	A2	A8
	FC	DC	39 9	36 6	33 3	8F	B2	B8
D	ED	CD	90	93	82	95	A3	A9
	FD	DD	90	93	82	95	B3	B9
	ED	CD	90	93	82 *	95	A3 *	A9 *
	FD	DD	90	93	82 *	95	B3 *	B9 *
E	EE	CE	91	80	84	81	A4	AA
	FE	DE	91	80	84	81	B4	BA
	EE	CE	91	80 *	84	81 *	A4 *	AA *
	FE	DE	91	80 *	84	81 *	B4 *	BA *
F	EF	CF	92	94	83	96	A5	AB
	FF	DF	92	94	83	96	B5	BB
	EF	CF	92	94	83 *	96	A5 *	AB *
	FF	DF	92	94	83 *	96	B5 *	BB *

This row contains the latched keys when that mode is selected (KMR6, KMR5 = 00)



TB00320S

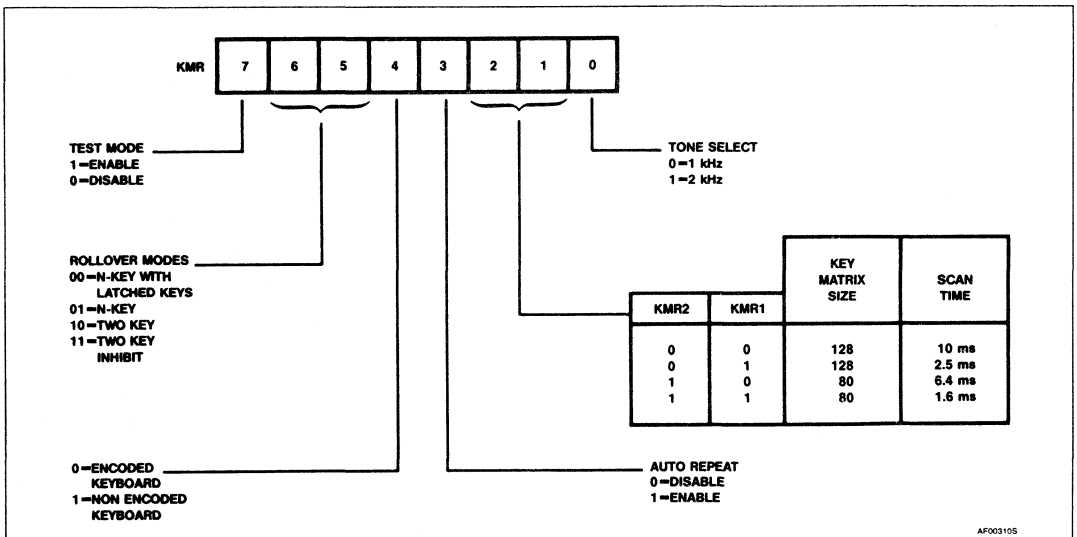


Figure 3. Keyboard Mode Register Format

Bit KMR3 enables the auto-repeat mode. In this mode, if a key that is programmed for auto-repeat is depressed for longer than one-half second, the key code will be loaded into the KDR approximately 15 times per second until that key is released. Only the non-control key codes will auto-repeat, i.e. $\overline{\text{CONTROL}}$ = 1. Table 2 specifies the auto-repeat keys.

KMR2 and KMR1 select the key matrix size and debounce time (scan rate). The keyboard row outputs (KR2, KR1, KR0) always scan from 0 to 7. The column outputs (KC3, KC2, KC1, KC0) scan from 0 to 15 for a 128 key matrix and from 0 to 9 for an 80 key matrix.

KMR0 selects between a 1kHz and 2kHz frequency to be output on the TONE pin in response to a ring tone command.

Keyboard Status Register

The keyboard status register (KSR) provides operational feedback to the CPU. Its format is illustrated in figure 4.

KSR7, 6 and 4 reflect the state of the inputs at the corresponding pins. $\overline{\text{CONTROL}}$ and $\overline{\text{SHIFT}}$ are latched at the time the key is accepted. As the verified codes are loaded into the KDR, the corresponding states of $\overline{\text{CONTROL}}$ and $\overline{\text{SHIFT}}$ are loaded into the KSR. $\overline{\text{REPEAT}}$ is updated on every matrix sample. The status bits are the complements of the input levels.

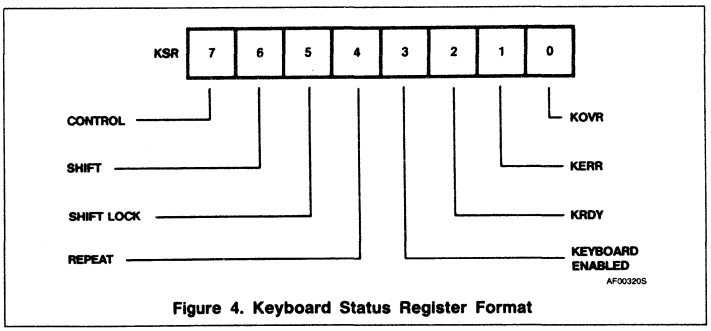


Figure 4. Keyboard Status Register Format

KSR5 reflects the state of the internal shift lock flag which is controlled by the set/reset shift lock commands.

KSR3 indicates that the keyboard controller is enabled. It is controlled by the set/clear keyboard enable command.

Keyboard overrun (KSR2) is set when both the KHR and KDR are full and a third key is validated. The original content of the KHR is preserved and the content of the KDR is overwritten with the new key code. This bit can be specified (by IMR1) to generate an interrupt and is cleared by the reset command with D2 = 1.

Keyboard error (KSR1) is set when the operator depresses more keys than are allowed in the selected rollover mode, or when keys are depressed simultaneously (within one scan cycle). This bit can be specified (by IMR3) to generate an interrupt and is cleared by the reset command with D1 = 1.

Keyboard data ready (KSR0) is set when the key code or address is transferred from the KDR to the KHR. This bit can be specified (by IMR2) to generate an interrupt. It is cleared when the CPU reads the KHR.

Communications Controller

The communications controller section of the PKCC comprises a full duplex asynchronous receiver/transmitter (UART) with a baud rate generator. Registers associated with these elements are the communications mode register (CMR), the baud rate control register (BRR), and the communications status register (CSR).

Receiver

The receiver accepts serial data on the RxD pin, converts the serial input to parallel format, checks for start bit, stop bit, parity bit (if any), or break condition, and presents the assembled character to the CPU. The receiver looks for a high to low (mark to space) transition of the start bit on the RxD input pin. If a transition is detected, the state of the RxD pin is sampled again after a delay of one half of the bit time. If RxD is then high, the start bit is invalid and the search for a valid start bit begins again. If RxD is still low, a valid start bit is assumed and the receiver continues to sample the input at one bit time intervals at the theoretical center of the bit, until the proper number of data bits and the parity bit (if any) have been assembled, and one stop bit has been detected. The least significant bit is received first. The data is then transferred to the receive holding register (RxHR) and the RxRDY bit in the CSR is set to a 1. If the character length is less than eight bits, the most significant unused bits in the RxHR are set to zero.

After the stop bit is detected, the receiver will immediately look for the next start bit. However, if a non-zero character was received without a stop bit (i.e. framing error) and RxD remains low for one half of the bit period after the stop bit was sampled, then the space is interpreted as a start bit.

The parity error, framing error and overrun error (if any) are strobed into the CSR at the received character boundary. If a break condition is detected (RxD is low for the entire character including the stop bit) only one character consisting of all zeros will be transferred to the RxHR and the received break bit in the CSR is set to 1 (RxRDY is not set when a break is received). The RxD input must return to a high condition for one bit time before a search for the next start bit begins.

Transmitter

The transmitter accepts parallel data from the CPU and converts it to a serial bit stream on the TxD output pin. It automatically sends a start bit followed by the data bits, an optional parity bit, and the programmed number of stop bits. The least significant bit is sent first. Following the transmission of the stop bits, if a new character is not available in the transmit holding register (TxHR), the TxD output remains high and the TxEMT bit in the CSR will be set to 1. Transmission resumes and the TxEMT bit is cleared when the CPU loads a new character into the TxHR. The transmitter can be forced to send a continuous low condition by a transmit break command.

If the transmitter is disabled, it continues operating until the character currently being transmitted is completely sent out.

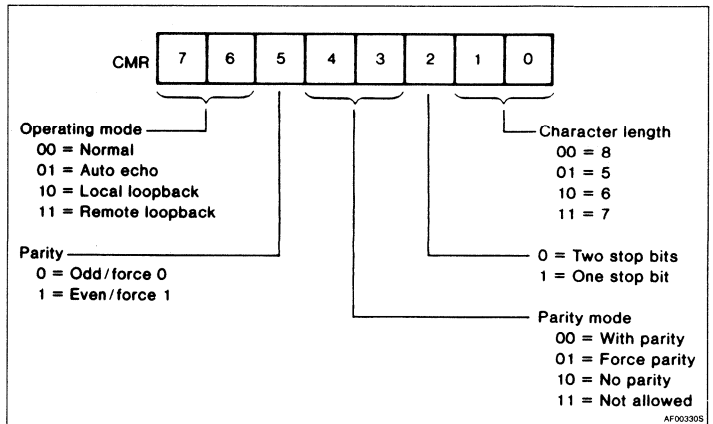


Figure 5. Communications Mode Register Format

Communication Mode Register

Figure 5 illustrates the bit format of the CMR, which controls the operational mode of the communications controller and the character parameters.

Bits CMR1 – CMR0 select a character length of 5, 6, 7, or 8 bits. The character length does not include the parity, start, or stop bits.

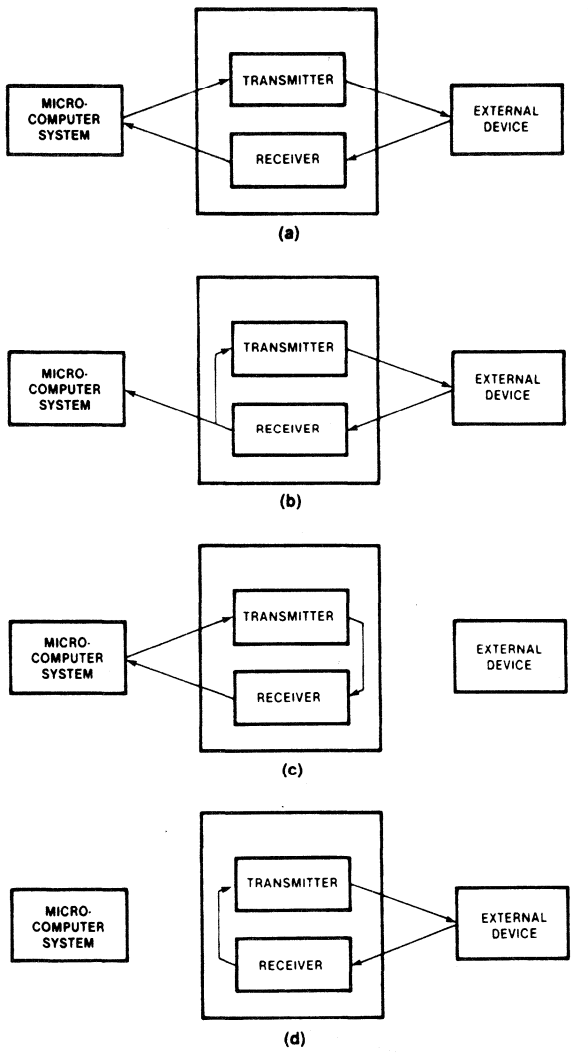
CMR2 selects the transmitted character framing as one or two stop bits. The receiver always checks for one stop bit.

The parity format is selected by bits CMR4 and CMR3. If parity or force parity is selected, a parity bit is added to the transmitted character and the receiver performs a parity check on incoming data. CMR5 selects odd or even parity and determines the polarity of the parity bit in the force parity mode.

The bits in the mode register affecting character assembly and disassembly (CMR5 – CMR0) can be changed dynamically and affect the characters currently being assembled in RxSR and transmitted by TxSR. To affect assembly of a received character, the CMR must be updated within n–1 bit times of the receipt of that character's start bit. To affect a transmitted character, the CMR must be updated within n–1 bit times of transmitting that character's start bit. (n = the smaller of the new and old character lengths).

The UART can operate in one of four modes, as illustrated in figure 6. The operating modes are selected by bits CMR7 and CMR6, which should only be changed when both the transmitter and receiver are disabled. CMR7-CMR6 = 00 is the normal mode, with the transmitter and receiver operating independently. CMR7-CMR6 = 01 places the UART in the automatic echo mode, which automatically retransmits the received data. The following conditions are true while in automatic echo mode:

1. Data assembled by the receiver is automatically placed in the transmit holding register and retransmitted on the TxD output.
2. The receive clock is used for the transmitter.
3. The receiver must be enabled, but the transmitter need not be enabled.
4. Status bit TxRDY is not set. TxEMT operates normally.
5. The receiver parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
6. Only the first character of a break condition is echoed; the TxD output will go high until the next received character is assembled.
7. CPU to receiver communication continues normally, but the CPU to transmitter link is disabled.



NOTES:
(a) Normal operating mode.
(b) Automatic echo mode.
(c) Local loopback mode.
(d) Remote loopback mode.

Figure 6. Operating Modes of the 2671 UART

Two diagnostic modes can also be configured. In local loopback mode (CMR7-CMR6 = 10):

1. The transmitter output is internally connected to the receiver input.
2. The transmit clock is used for the receiver.
3. The TxD output is held high.
4. The RxD input is ignored.
5. The transmitter must be enabled, but the receiver need not be enabled.
6. CPU to transmitter and receiver communications continue normally.

The second diagnostic mode is the remote loopback mode (CMR7-CMR6 = 11). In this mode:

1. Data assembled by the receiver is automatically placed in the transmit holding register and retransmitted on the TxD output.
2. The receive clock is used for the transmitter.
3. No data is sent to the local CPU, but the error status conditions (parity and framing) are set if required.
4. The received parity is checked, but is not regenerated for transmission, i.e., transmitted parity bit is as received.
5. The receiver must be enabled, but the transmitter need not be enabled.

Baud Rate Control Register

The baud rate control register (BRR) controls the frequency generated by the baud rate generator (BRG) and the clock source used by the receiver and transmitter. Its format is illustrated in figure 7.

BRR3 – BRR0 select one of sixteen frequencies to be generated by the BRG. See table 3.

BRR7 and BRR6 select the source of the transmit and receive clocks. If external clocks are chosen, (BRR7 = 0 or BRR6 = 0), then the clock rate factor is determined by BRR5 and BRR4. The external clock input(s) should be the desired baud rate multiplied by the clock rate factor.

If internal clock(s) are specified, (BRR7 = 1 or BRR6 = 1), the clock is supplied by the internal baud rate generator at the selected baud rate. The clock rate factor for internally generated clocks is always 16. Pins 35 and 34 become outputs for transmit or receive clocks, respectively. See table 4 for the description and selection of these outputs.

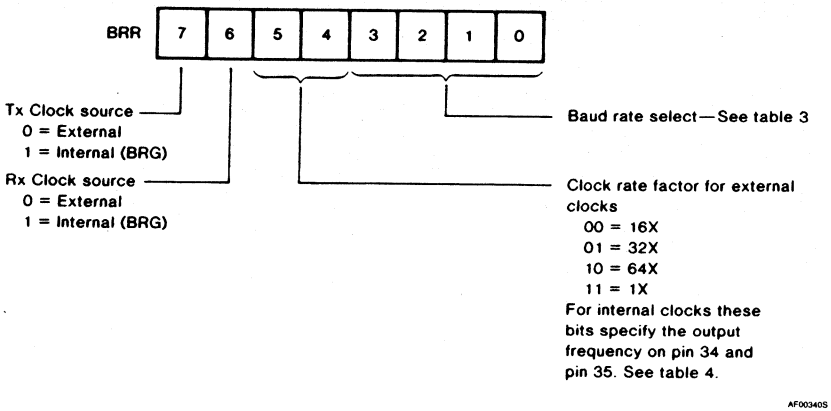


Figure 7. Baud Rate Control Register Format

Table 3. BAUD RATE GENERATOR CHARACTERISTICS (BRCLK = 4.9152MHz)

BRR3-0	BAUD RATE	ACTUAL FREQUENCY 16X CLOCK	PERCENT ERROR	DIVISOR
0000	50	0.8 kHz	–	6144
0001	110	1.7598	-0.01	2793
0010	134.5	2.152	–	2284
0011	150	2.4	–	2048
0100	200	3.2	–	1536
0101	300	4.8	–	1024
0110	600	9.6	–	512
0111	1050	16.8329	+0.20	292
1000	1200	19.2	–	256
1001	1800	28.7438	-0.20	171
1010	2000	31.9168	-0.26	154
1011	2400	38.4	–	128
1100	4800	76.8	–	64
1101	9600	153.6	–	32
1110	19200	307.2	–	16
1111	38400	614.4	–	8

Table 4. BAUD RATE CONTROL REGISTER

BRR7 – BRR4	CLOCK SOURCE		PIN FUNCTIONS		BRR3 – BRR0 BAUD RATE SELECTION
	TxC	RxC	PIN 34	PIN 35	
00**	E	E	TxC	RxC	The baud rates are listed in table 3.
01**	E	I	TxC	1X	
10**	I	E	16X	RxC	
1100	I	I	1X	1X	
1101	I	I	1X	16X	
1110	I	I	16X	1X	
1111	I	I	16X	16X	

NOTES:
1. ** = Clock rate factor for external clocks: 00 = 16X
   01 = 32X
   10 = 64X
   11 = 1X
2. E = External clock.
3. I = Internal clock (BRG).
4. 1X and 16X are clock outputs at 1 or 16 times the actual baud rate. For receive, the 1X output is the actual data sample clock.
5. BRR7 – BRR6 = 01 or 10 not permitted in automatic echo or remote loopback modes unless BRR5 – BRR4 = 00.

Communications Status Register

Figure 8 illustrates the bit format of the communications status register (CSR), which provides UART status to the CPU.

Receiver ready (CSR0) indicates that a received character is assembled and transferred to the RxHR and is ready to be read by the CPU. This bit can be specified (by IMR0) to generate an interrupt and is reset by reading the RxHR.

Transmitter ready (CSR1) indicates that the TxHR is empty and ready to be loaded with a character. This bit will be cleared when the TxHR is loaded and has not yet transferred the character to the transmit shift register (TxSR). TxRDY is reset when the transmitter is disabled. It will be set when the transmitter is enabled, provided that no data was loaded into the TxHR during the time the transmitter was disabled. This bit can be specified (by IMR7) to generate an interrupt.

Transmitter empty (CSR2) indicates that the transmitter has underrun, i.e., both the TxHR and TxSR are empty. This bit can only be set after transmission of at least one character, and is cleared when the TxHR is loaded by the CPU. TxEMT is reset when the transmitter is disabled. This bit can be specified (by IMR6) to generate an interrupt.

CSR3 will be set when the PKCC receives a command to transmit a break. This bit will be cleared after the break is completed.

Received break (CSR4) indicates that an all zero character of the programmed length has been received without a stop bit. Breaks originating in the middle of a received character can be detected. This bit is cleared when RxD returns to a high state for at least one bit time.

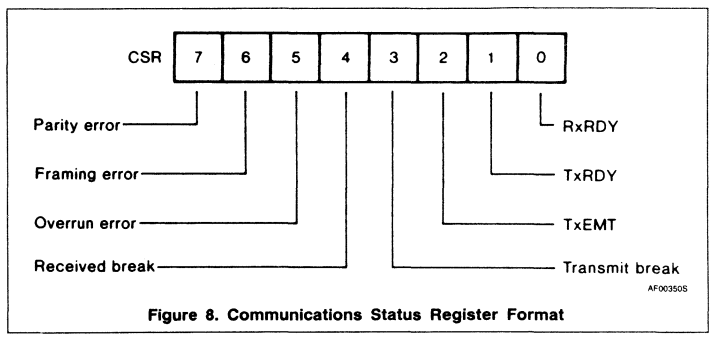


Figure 8. Communications Status Register Format

Receiver overrun (CSR5) indicates that the previous character in the RxHR has not been read by the CPU and that a new character has been loaded into the RxHR. This bit is cleared by a reset command with D3 = 1.

Framing error (CSR6) indicates that the stop bit has not been detected. The stop bit check is made in the middle of the first stop bit position. This bit is cleared by a reset command with D3 = 1.

Parity error (CSR7) indicates that a character was received with incorrect parity when 'with parity' or 'force parity' is enabled. This bit is cleared by a reset command with D3 = 1.

Interrupt Controller

The SCN2671 contains a maskable interrupt status register (ISR) which can be enabled to generate an active low interrupt request on the $\overline{INTR}$ output. The eight interrupt conditions in the ISR are individually enabled by writing a 1 into the corresponding bit of the interrupt mask register (IMR).

Each of the interrupt conditions is assigned a priority and a vector. When an enabled ISR bit is set, the SCN2671 asserts the $\overline{INTR}$ output. If the CPU activates the $\overline{INTA}$ input, the SCN2671 responds by placing the corresponding 8-bit vector on the data bus (D7 – D0). If multiple interrupts are pending, the vector corresponds to the condition with the highest priority. The interrupt will persist until all pending interrupt conditions are cleared.

The ISR can also be polled by reading at address A2 – A0 = 000. All pending interrupt conditions which are enabled by the IMR will be read independent of priority.

The bit assignments of the ISR and IMR and corresponding vectors and priorities are listed in table 5.

Table 5. INTERRUPT MASK REGISTER (IMR) AND INTERRUPT STATUS REGISTER (ISR)

BIT IN IMR/ISR	INTERRUPT CONDITION	PRIORITY	VECTOR ON D7 - D0		CONDITION RESET BY:
			BINARY	HEX	
IMR0/ISR0	RxRDY	1	11001111	CF	Read RxHR
IMR1/ISR1	KOVR	2	11010111	D7	Reset CMD (D2 = 1)
IMR2/ISR2	KRDY	3	11011111	DF	Read KHR
IMR3/ISR3	KERR	4	11100111	E7	Reset CMD (D1 = 1)
IMR4/ISR4	XINT[1]	5	11101111	EF	External
IMR5/ISR5	△BREAK[2]	6	11110111	F7	Reset CMD (D4 = 1)
IMR6/ISR6	TxEMT	7	11000111	C7	Load TxHR
IMR7/ISR7	TxRDY	8	11000111	C7	Load TxHR

NOTES:
1. XINT is an input from an external interrupt source, active low (pin 21).
2. △BREAK refers to the change of a received break condition.

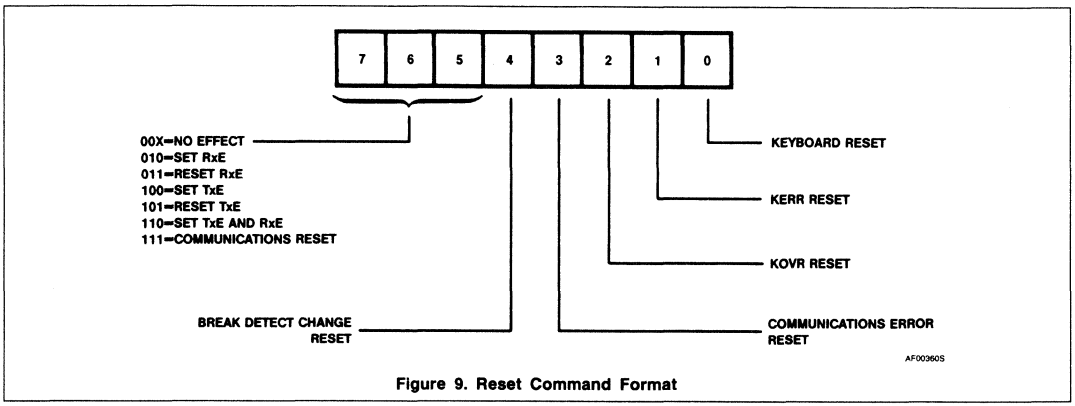


Figure 9. Reset Command Format

COMMANDS

In addition to the control exercised by programming of the PKCC control registers, several functions can be performed by executing command operations. There are two classes of commands which are initiated by writing to the SCN2671 at address A2 - A0 = 000 (reset command) and address A2 - A0 = 111 (miscellaneous commands). Individual commands are specified by the bit pattern on the data bus (D7 - D0).

Reset Commands

The reset command bit format is illustrated in figure 9 and the detail command descriptions are given in table 6.

A reset command with D7 - D0 = 111XXXX1 is a master reset for the SCN2671. This command must be given following a power on condition to release the internal power on reset latch which deactivates the SCN2671 on power up.

Miscellaneous Commands

The miscellaneous command format is illustrated in figure 10.

The transmit break commands force a break (steady low output) on the TxD pin immediately or after the character in the TxSR (if any) is transmitted. A timed break lasts for approximately 200ms, and a character break lasts for one character time including parity and stop bit time. In either case, TxRDY (CSR1) will be set at the beginning of the break which can be extended indefinitely (by 200ms or one character time increments) by reasserting the command in response to TxRDY. Note that these commands reset TxRDY. When a transmit break command is asserted, CSR3 will be set. The bit will be cleared after the break is completed.

The ring tone commands cause the tone generator to output a square wave on the TONE output. The tone durations are specified by the commands:

Ring tone short = 25ms
Ring tone long = 100ms

The tone frequency is either 1kHz or 2kHz, as specified by KMR0.

Table 6. RESET COMMAND DESCRIPTION

COMMAND	RESETS	COMMENTS
Keyboard reset	KMR7 – KMR0 KSR5, KSR3 – KSR0 IMR3 – IMR1	The keyboard controller is reset, ignoring the input at KRET.
KERR reset	KSR1	Keyboard error status bit reset.
KOVR reset	KSR2	Keyboard overrun status bit reset.
Communications error reset	CSR7-CSR5	Resets the receiver overrun, parity, and framing error status bits.
Break detect change reset	ISR5	Resets the break detect change bit in the interrupt status register.
Set RxE	See note.	Enables receiver operation.
Reset RxE	CSR7 – CSR4, CSR0 See note.	Disables the receiver.
Set TxE	See note.	Enables transmitter operation.
Reset TxE	CSR3-CSR1 See note.	Disables the transmitter. Sets the TxD output to a 1 after transmitting the character in TxSR.
Communications reset	CMR, CSR, BRR, TxE, RxE, IMR7-IMR5, IMR0	Resets the communication controller. The RxD input is ignored and the TxD output is set to a 1.
Master reset	CMR, CSR, BRR, TxE, RxE, KMR, KSR5, KSR3-KSR0, IMR7-IMR0 Releases the internally latched power on reset.	Resets the keyboard and communication controllers. Inputs at KRET and RxD are ignored and the TxD output is set to a 1.

NOTE:
Command does not affect the CMR or the BRR.

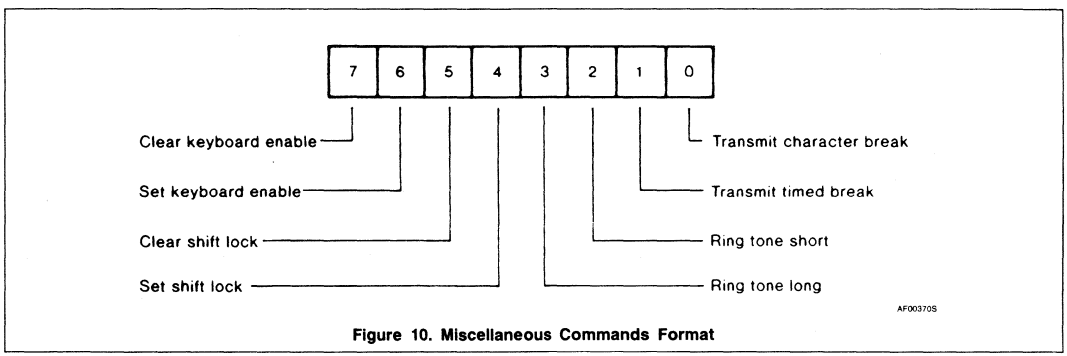


Figure 10. Miscellaneous Commands Format

The set/clear shift lock commands control the state of the internal shift lock flip flop. When shift lock is set, the keyboard controller encodes all key depressions as if the $\overline{\text{SHIFT}}$ input was asserted. The state of the shift lock flip flop is reflected in KSR5.

The set keyboard enable command enables the keyboard controller and sets KSR3 in the keyboard status register. THe clear keyboard enable command resets KSR3 and disables key processing at the KRET input. The keyboard controller is not reset by this command, and the current state of the keyboard (key depressions and latched key states) is preserved internally. When the keyboard is subsequently enabled, key processing resumes, old and new keys are debounced, and latched keys are encoded if there has been a change in their state.

MASK PROGRAMMABLE OPTIONS

Characteristics of certain portions of the PKCC are internally programmed by means of a read only memory. The items which can be programmed are:

- Key codes
- Auto-repeat keys
- Scan times, tone frequency, and tone duration
- Baud rates
- Interrupt vectors

Consult your local Signetics representative for costs, minimum quantities, and data submission requirements for customized versions of the PKCC.

ABSOLUTE MAXIMUM RATINGS

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$[4,5,6]

	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage					
	XTAL1, XTAL2/BRCLK		4.0			V
	All other inputs		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 1.6mA$			0.4	V
V_{OH}	Output high voltage (except $\overline{INTR}$)	$I_{OH} = -100\mu A$	2.4			V
I_{IL}	Input leakage current (except XTAL1, XTAL2/BRCLK)	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{XTLIL}	Input low current	$V_{IN} = 0$				
	XTAL1		−80	−30		μA
	XTAL2/BRCLK[7]		−4.0	−1.5		mA
I_{XTLIH}	Input high current	$V_{IN} = V_{CC}$				
	XTAL1			30	80	μA
	XTAL2/BRCLK[7]			0.2	1.0	mA
I_{LL}	Data bus 3-state leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{CC}	Power supply current				150	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5V \pm 5\%$[4,5,6]

	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
Read timing (See figure 11)						
t_{AS}	Address set-up to $\overline{RD}$		50			ns
t_{CS}	$\overline{CE}$ set-up to $\overline{RD}$		50			ns
t_{PW}	$\overline{RD}$ pulse width		250			ns
t_{AH}	Address hold from $\overline{RD}$		20			ns
t_{CH}	$\overline{CE}$ hold from $\overline{RD}$	$C_L = 150pF$	0			ns
t_{DD}	Data delay for read				200	ns
t_{DF}	Data bus floating time for read	$C_L = 150pF$	10		100	ns
t_{AD1}	Access delay from any read to next read or write		250			ns
Write timing (See figure 12)						
t_{AS}	Address setting to $\overline{WR}$		50			ns
t_{CS}	$\overline{CE}$ set-up to $\overline{WR}$		50			ns
t_{PW}	$\overline{WR}$ pulse width		250			ns
t_{AH}	Address hold from $\overline{WR}$		20			ns
t_{CH}	$\overline{CE}$ hold from $\overline{WR}$		0			ns
t_{DS}	Data set-up		100			ns
t_{DH}	Data hold		10			ns
t_{AD2}	Access delay from any write to next read or write		250			ns
	Access delay from reset command to next read or write		1.0			μs

AC ELECTRICAL CHARACTERISTICS (Continued)

	PARAMETER	TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
	Interrupt acknowledge timing (see figure 13)					
t_{PWI}	$\overline{INTA}$ pulse width		300			ns
t_{DDI}	Data delay time for interrupt vector	C_L = 150pF			250	ns
t_{DFI}	Data bus floating time after $\overline{INTA}$	C_L = 150pF	10		100	ns
t_{ADI}	$\overline{INTA}$ to $\overline{INTA}$ access delay		300			ns
	$\overline{INTR}$ reset timing (see figure 14)					
t_{RI}	$\overline{INTR}$ delay from:					
	Read RxHR (RxRDY)				400	ns
	Read KHR (KRDY)				400	ns
	Reset commands (KOVR,KERR,BREAK)				450	ns
	Load TxHR (TxEMT,TxRDY)				400	ns
	Mask bit reset				300	ns
	Keyboard timing (see figure 15 and 16)					
f_{KCLK}	KCLK frequency			409		kHz
t_{KBD}	KR_i,KC_i to KRET sample delay:					
	FAST SCAN		12.0			µs
	SLOW SCAN		55.0			µs
t_{POS}	Scan time per matrix position:					
	FAST SCAN			20		µs
	SLOW SCAN			80		µs
t_{KRD}	KDRES delay from KCLK	C_L = 150pF			400	ns
t_{KRH}	KDRES hold from KLCK	C_L = 150pF			400	ns
t_{HYSD}	$\overline{HYS}$ delay from KCLK	C_L = 150pF			600	ns
t_{RCD}	KR_i,KC_i delay from KLCK	C_L = 150pF			400	ns
	UART timing (see figure 17, 18, 19)					
t_{RXS}	RxD set-up time		200			ns
t_{RxH}	RxD hold time		200			ns
t_{TxD}	TxD delay from falling edge of TxC	C_L = 150pF			300	ns
t_{TCS}	Skew between TxD transition and falling edge of TxC output	C_L = 150pF		0		ns
t_{BRH}	XTAL1 clock high[8]		70			ns
t_{BRL}	XTAL1 clock low[8]		70			ns
f_{BRG}	BRG input frequency		1.0	4.9152	5.075	MHz
$f_{R/T}$	TxC or RxC input frequency	Clock rate factor = 16X, 32X, 64X			1.3	MHz
		Clock rate factor = 1X			1.0	MHz
$t_{R/TH}$	TxC or RxC clock high		350			ns
$t_{R/TL}$	TxC or RxC clock low		350			ns

NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is stress rating only and functional operation of the device at these or at any other condition above those indicated in the operation section of this specification is not implied.
2. For operating on elevated temperatures, the device must be operated based on +150°C maximum function temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over operating temperature range unless otherwise specified.
5. All voltage measurements are referenced to ground (V_{SS}). All input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum and time measurements are referenced at input voltages of 0.8V, 2.0V and at output voltages of 0.8V, 2.0V as appropriate, unless otherwise specified.
6. Typical values are at +25°C, typical supply voltages and typical processing parameters.
7. XTAL2 input currents are measured with XTAL grounded.
8. See figures 20 and 21 for XTAL1, WTAL2 connections for driving XTAL2 with an external clock. Input levels for XTAL1 and XTAL2 are $V_{IL} \leq 0.8V$, $V_{IH} \geq 4.0V$, and t_{BRL} and t_{BRH} are measured at these levels.

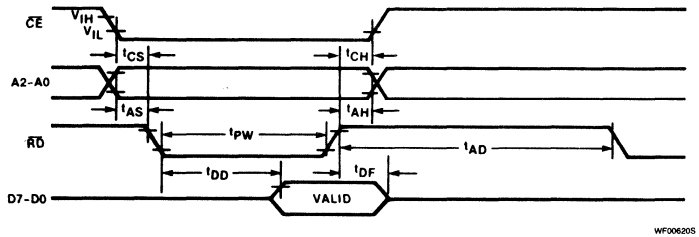


Figure 11. Read Timing

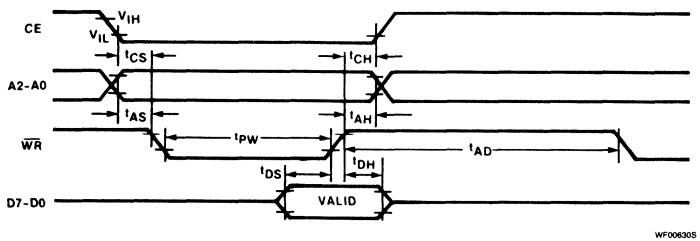


Figure 12. Write Timing

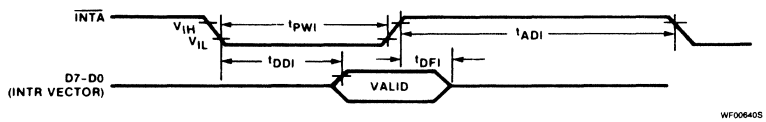


Figure 13. Interrupt Acknowledge Timing

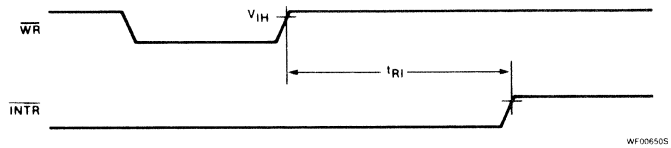
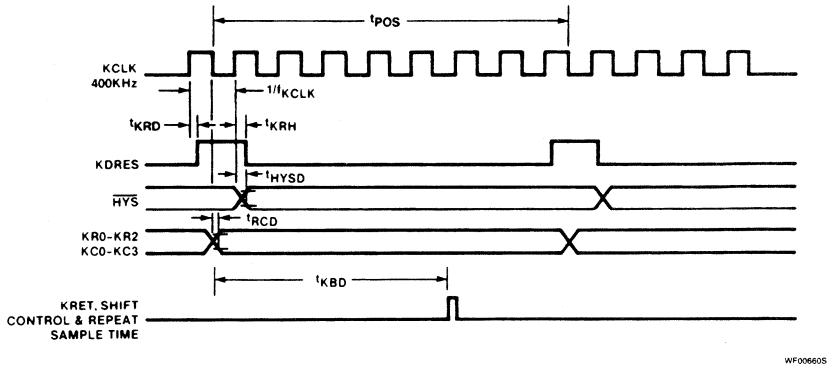


Figure 14. Interrupt Reset Timing



NOTE:
Scan timing shown is for fast scan (KMR1 = 1). For slow scan (KMR1 = 0) all signals except KCLK run at 1/4 the shown rates.

Figure 15. Keyboard Scan Timing

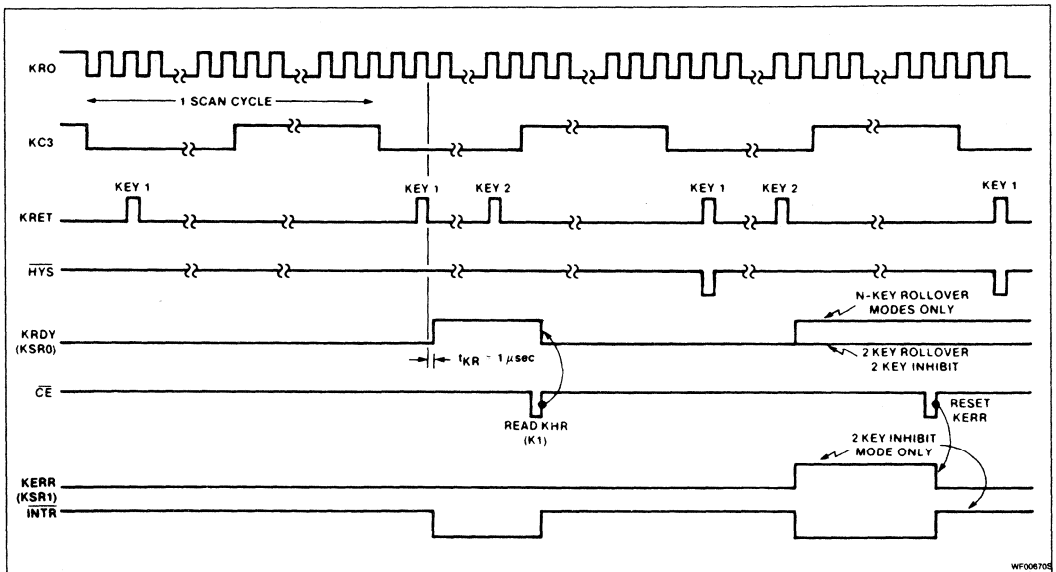


Figure 16. Keyboard Timing

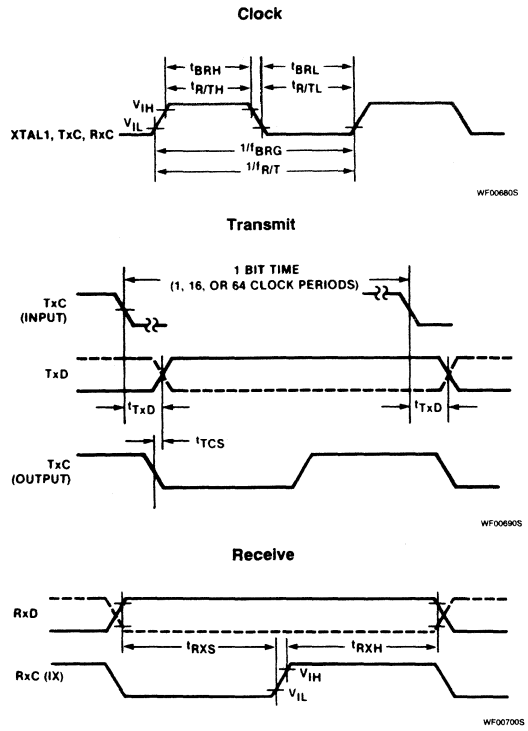

Clock
tBRH
tR/TH
tBRL
tR/TL
VIH
VIL
XTAL1, TxC, RxC
1/fBRG
1/fR/T
WF00680S
Transmit
1 BIT TIME
(1, 16, OR 64 CLOCK PERIODS)
TxC
(INPUT)
TxD
tTxD
tTxD
tTCS
TxC
(OUTPUT)
WF00690S
Receive
RxD
tRXS
tRXH
RxC (IX)
VIH
VIL
WF00700S

Figure 17.

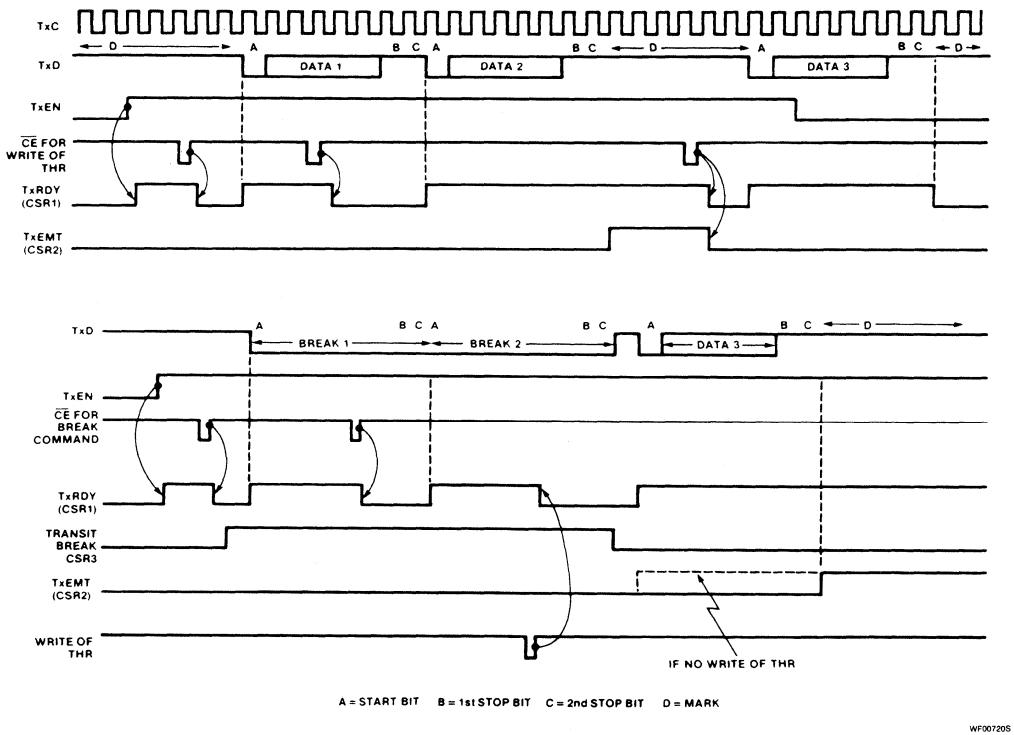


Figure 18. Transmitter Timing (5-Bit Characters, no Parity, 2 Stop Bits)

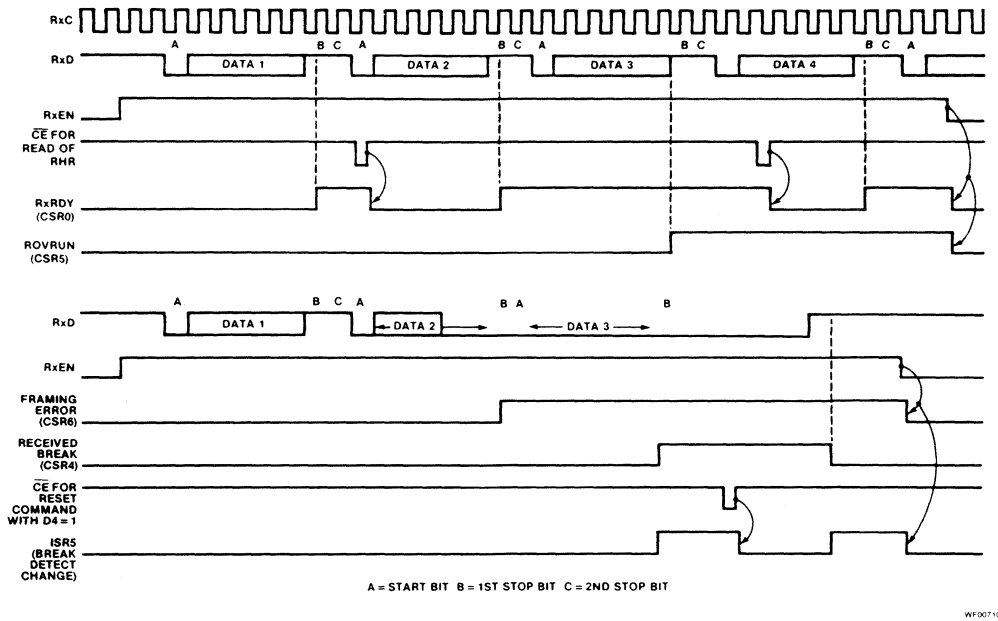


Figure 19. Receiver Timing (5-Bit Characters, no Parity, 2 Stop Bits)

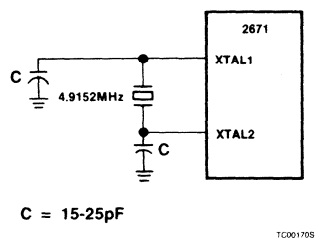


Figure 20. Crystal Connections BRG Clock

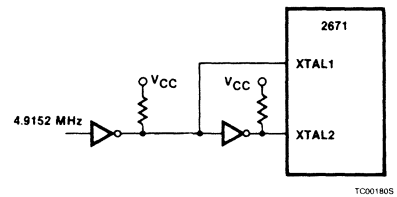


Figure 21. Connection for External BRG Clock Source

Table 7. REGISTER FORMAT SUMMARY

	7	6	5	4	3	2	1	0
KMR	Test Mode 1 = Enable 0 = Disable	Rollover modes 00 = N-key with latched keys 01 = N-key 10 = Two keys 11 = Two key inhibit		Keyboard 0 = Encoded 1 = Nonen-coded	Auto repeat 0 = Disable 1 = Enable	KMR2 / KMR1 0 0 0 1 1 0 1 1	Key Matrix Size 128 128 80 80	Scan Time 10ms 2.5ms 6.4ms 1.6ms	Tone select 0 = 1kHz 1 = 2kHz
KSR	CONTROL	SHIFT	SHIFT LOCK	REPEAT	Keyboard Enabled	KOVR	KERR	KRDY	
CMR	Operating Mode 00 = Normal 01 = Auto echo 10 = Local loopback 11 = Remote loopback		Parity 0 = Odd/ force 0 1 = Even/ force 1	Parity Mode 00 = With parity 01 = Force parity 10 = No parity 11 = Not allowed	Stop Bits 0 = Two 1 = One	Character Length 00 = 8 01 = 5 10 = 6 11 = 7			
BRR	Tx Clock source 0 = External 1 = Internal (BRG)	Rx Clock source 0 = External 1 = Internal (BRG)	Clock rate factor for external clocks 00 = 16X 01 = 32X 10 = 64X 11 = 1X For internal clocks these bits specify the output frequency on pins 34 and 35 (table 4).	Baud rate select (BRR3-BRR0 in hex) 0 = 50 1 = 110 2 = 134.5 3 = 150 (BRCLK = 4.9152MHz)	4 = 200 5 = 300 6 = 600 7 = 1050	8 = 1200 9 = 1800 A = 2000 B = 2400	C = 4800 D = 9600 E = 19200 F = 38400		
CSR	Parity error	Framing error	Overrun error	Received break	Transmit break	TxEMT	TxRDY	RxRDY	
IMR/ISR	TxRDY	TxEMT	BREAK CHANGE	XINT	KERR	KRDY	KOVR	RxRDY	
Reset Command Format	00X = No effect 010 = Set RxE 011 = Reset RxE 100 = Set TxE	101 = Reset TxE 110 = Set TxE and RxE 111 = Communica-tions reset		Break detect change reset	Communi-cations error reset	KOVR reset	KERR reset	Keyboard reset	
Miscellaneous Commands Format	Clear keyboard enable	Set keyboard enable	Clear shift lock	Set shift lock	Ring tone long	Ring tone short	Transmit timed break	Transmit character break	

Programmable Video Timing Controller (PVTC)

Product Specification

Originally published by Signetics February 1985

DESCRIPTION

The Signetics SCN2672 Programmable Video Timing Controller (PVTC) is a programmable device designed for use in CRT terminals and display systems that employ raster scan techniques. The PVTC generates the vertical and horizontal timing signals necessary for the display of interlaced or non-interlaced data on a CRT monitor. It provides consecutive addressing to a user specified display buffer memory domain and controls the CPU-display buffer interface for various buffer configuration modes. A variety of operating modes, display formats, and timing profiles can be implemented by programming the control registers in the PVTC.

A minimum CRT terminal system configuration consists of a PVTC, an SCN2671 Keyboard and Communication Controller (PKCC), an SCN2670 Display Character and Graphics Generator (DCGG), an SCN2673/2677 Video and Attributes Controller (VAC), a single chip microcomputer such as the 8048, a display buffer RAM, and a small amount of TTL for miscellaneous address decoding, interface, and control. Typically, the package count for a minimum system is between 15 and 20 devices; system complexity can be enhanced by upgrading the microprocessor and expanding via the system address and data busses.

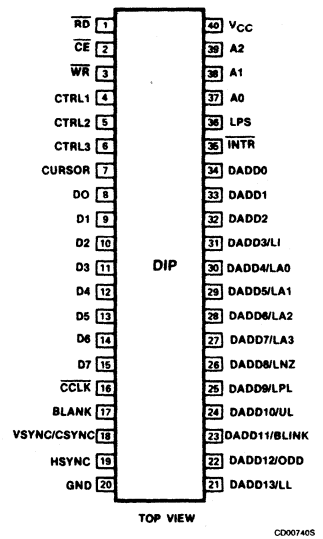
FEATURES

- **4MHz and 2.7MHz character rate versions**
- **Up to 256 characters per row**
- **1 to 16 raster lines per character row**
- **Up to 128 character rows per frame**
- **Programmable horizontal and vertical sync generators**
- **Interlaced or non-interlaced operation**
- **Up to 16K RAM addressing for multiple page operation**
- **Automatic wraparound of RAM**
- **Addressable incrementable and readable cursor**
- **Programmable cursor size, position, and blink**
- **Split screen and horizontal scroll capability**
- **Light pen register**
- **Selectable buffer interface modes**
- **Dynamic RAM refresh**
- **Completely TTL compatible**
- **Single +5 volt power supply**
- **Power on reset circuit**

APPLICATIONS

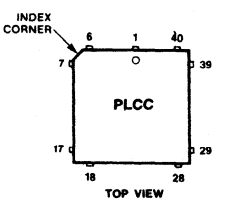
- **CRT terminals**
- **Word processing systems**
- **Small business computers**
- **Home Computers**

PIN CONFIGURATION



TOP VIEW

CD00740S



TOP VIEW

CD0044PS

Pin	Function	Pin	Function
1	NC	23	NC
2	$\overline{RD}$	24	DADD13/LL
3	$\overline{CE}$	25	DADD12/ODD
4	$\overline{WR}$	26	DADD11/BLINK
5	CTRL1	27	DADD10/UL
6	CTRL2	28	DADD9/LPL
7	CTRL3	29	DADD8/LNZ
8	CURSOR	30	DADD7/LA3
9	D0	31	DADD6/LA2
10	D1	32	DADD5/LA1
11	D2	33	DADD4/LA0
12	NC	34	NC
13	D3	35	DADD3/LI
14	D4	36	DADD2
15	D5	37	DADD1
16	D6	38	DADD0
17	D7	39	$\overline{INTR}$
18	$\overline{CCLK}$	40	LPS
19	BLANK	41	A0
20	VSYNC/CSYNC	42	A1
21	HSYNC	43	A2
22	GND	44	V_{CC}

ORDERING CODE

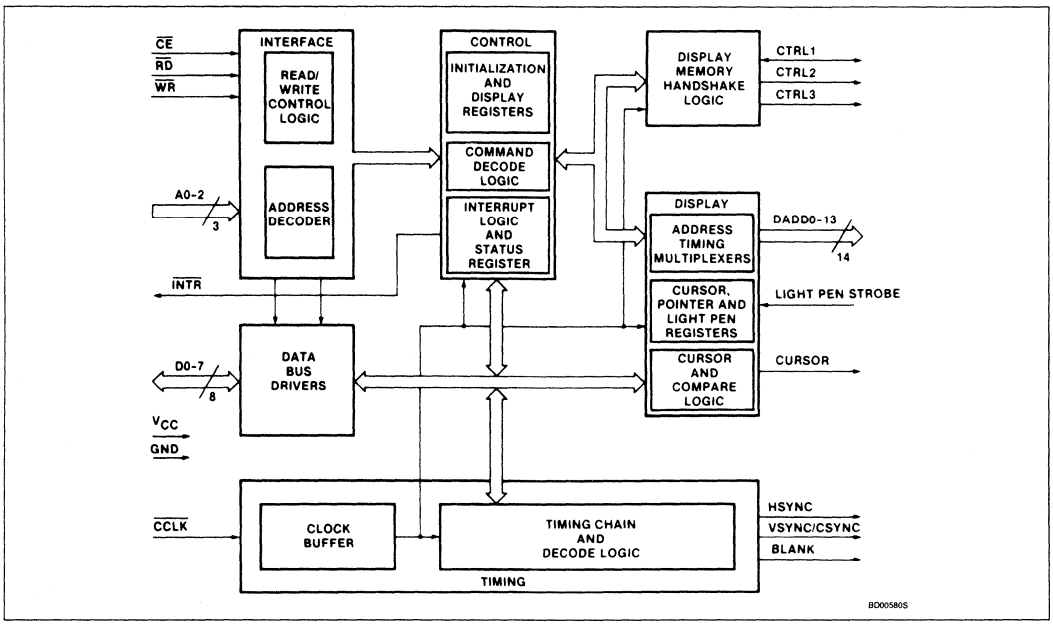
PACKAGES	V_{CC} = 5V ±5%, T_A = 0°C to 70°C	
	4MHz	2.7MHz
Ceramic DIP	SCN2672BC4I40	SCN2672BC3I40
Plastic DIP	SCN2672BC4N40	SCN2672BC3N40
Plastic LCC	SCN2672BC4A44	SCN2672BC3A44

FUNCTIONAL DESCRIPTION

As shown on the block diagram, the PVTC contains the following major blocks:

- Data bus buffer
- Interface Logic
- Operation Control
- Timing
- Display Control
- Buffer Control

BLOCK DIAGRAM



Data Bus Driver

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the PVTC.

Interface Logic

The interface logic contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The functions performed by the CPU read and write operations are as shown in table 1.

Table 1. PVTC ADDRESSING

A2	A1	A0	READ ($\overline{RD}$ = 0)	WRITE ($\overline{WR}$ = 0)
0	0	0	Interrupt register	Initialization registers[1]
0	0	1	Status register	Command register
0	1	0	Screen start address lower register	Screen start address lower reg.
0	1	1	Screen start address upper register	Screen start address upper reg.
1	0	0	Cursor address lower register	Cursor address lower register
1	0	1	Cursor address upper register	Cursor address upper register
1	1	0	Light pen address lower register	Display pointer address lower reg.
1	1	1	Light pen address upper register	Display pointer address upper reg.

NOTE:

1. There are 11 initialization registers which are accessed sequentially via a single address. The PVTC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR10, the split screen register) is accessed. The pointer then continues to point to the split screen register. Upon power-up or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command.

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
A0 – A2	37 – 39	41 – 43	I	**Address Lines:** Used to select PVTC internal registers for read/write operations and for commands.
D0 – D7	8 – 15	9 – 11, 13 – 17	I/O	**8-Bit Bidirectional Three-State Data Bus.** Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the PVTC take place over this bus. The direction of the transfer is controlled by the $\overline{RD}$ and $\overline{WR}$ inputs when the $\overline{CE}$ input is low. When the $\overline{CE}$ input is high, the data bus is in the three-state condition.
$\overline{RD}$	1	2	I	**Read Strobe:** Active low input. A low on this pin while $\overline{CE}$ is low causes the contents of the register selected by A0 – A2 to be placed on the data bus. The read cycle begins on the leading (falling) edge of $\overline{RD}$.
$\overline{WR}$	3	4	I	**Write Strobe:** Active low input. A low on this pin while $\overline{CE}$ is also low causes the contents of the data bus to be transferred to the register selected by A0 – A2. The transfer occurs on the trailing (rising) edge of $\overline{WR}$.
$\overline{CE}$	2	3	I	**Chip Enable:** Active low input. When low, data transfers between the CPU and the PVTC are enabled on D0 – D7 as controlled by the $\overline{WR}$, $\overline{RD}$, and A0 – A2 inputs. When $\overline{CE}$ is high, the PVTC is effectively isolated from the data bus and D0 – D7 are placed in the three-state condition.
$\overline{CCLK}$	16	18	I	**Character Clock:** Timing signal derived from the video dot clock which is used to synchronize the PVTC's timing functions.
HSYNC	19	21	O	**Horizontal Sync:** Active high output which provides video horizontal sync pulses. The timing parameters are programmable.
VSYNC/CSYNC	18	20	O	**Vertical Sync/Composite Sync:** A control bit selects either vertical or composite sync pulses on this active high output. When CSYNC is selected, equalization pulses are included. The timing parameters are programmable.
BLANK	17	19	O	**Blank:** This active high output defines the horizontal and vertical borders of the display. Display control signals which are output on DADD3 through DADD13 are valid on the trailing edge of BLANK.
CURSOR	7	8	O	**Cursor Gate:** This active high output becomes active for a specified number of scan lines when the address contained in the cursor registers match the address output on DADD0 through DADD13. The first and last lines of the cursor and a blink option are programmable.
$\overline{INTR}$	35	39	O	**Interrupt Request:** Open drain output which supplies an active low interrupt request from any of five maskable sources. This pin is inactive after power on reset or a master reset command.
LPS	36	40	I	**Light Pen Strobe:** Positive edge triggered input indicating a light pen hit. Causes the current value of the display address to be strobed into the light pen register.
CTRL1	4	5	I/O	**Handshake Control 1:** In independent mode, provides an active low write data buffer ($\overline{WDB}$) output which strobes data from the interface latch into the display memory. In transparent and shared modes, this is an active low processor bus request ($\overline{PBREQ}$) input which indicates that the CPU desires to access the display memory. This pin must be tied high when operating in row buffer mode.
CTRL2	5	6	O	**Handshake Control 2:** In independent mode, provides an active low read data buffer ($\overline{RDB}$) output which strobes data from the display memory into the interface latch. In transparent and shared modes, this is an active low bus external enable ($\overline{BEXT}$) output which indicates that the PVTC has relinquished control of the display memory (DADD0 – DADD13 are in the three-state condition) in response to a CPU bus request. $\overline{BEXT}$ also goes low in response to a 'display off and float DADD' command. In row buffer mode, it is an active low bus request ($\overline{BREQ}$) output which halts the CPU during a line DMA.
CTRL3	6	7	O	**Handshake Control 3:** In independent mode, provides the active low buffer chip enable ($\overline{BEC}$) signal to the display memory. In transparent and shared modes provides an active low bus acknowledge ($\overline{BACK}$) output which serves as a ready signal to the CPU in response to a processor bus request. In row buffer mode, this is an active high memory bus control

PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
DADD0 – DADD13	34 – 21	38 – 35 33 – 24	O	(MBC) output which configures the system for the DMA transfer of one row of character codes from system memory to the row display buffer. **Display Address:** Used by the PVTC to address up to 16K of display memory. These outputs are floated at various times depending on the buffer mode. Various control signals are multiplexed on DADD3 thru DADD13 and are valid at the trailing edge of BLANK. These control signals are: **DADD3 /LI** **Line Interlace:** Replaces DADD4/LA0 as the least significant line address for the interlaced sync and video applications. A low indicates an even row of an even field or an odd row of an odd field. **DADD4 – DADD7/LA0 – LA3** **Line Address:** Provides the number of the current scan line within each character row. **DADD8/LNZ** **Line Zero:** Asserted before the first scan line in each character row. **DADD9/LPL** **Light Pen Line:** Asserted before the scan line which matches the programmed light pen line position (line 3, 5, 7, or 9). **DADD10/UL** **Underline:** Asserted before the scan line which matches the programmed underline position (line 0 thru 15). **DADD11/BLINK** **Blink frequency:** Provides an output divided down from the vertical sync rate. **DADD12/ODD** **Odd Field:** Active high signal which is asserted before each scan line of the odd field when interlace is specified. **DADD13/LL** **Last Line:** Asserted before the last scan line of each character row.
V_{CC}	40	44	I	**Power Supply:** +5 volts ±5% power input.
GND	20	22	I	**Ground:** Signal and power ground input.

Operation Control

The operation control section decodes configuration and operation commands from the CPU and generates appropriate signals to other internal sections to control the overall device operation. It contains the timing and display registers which configure the display format and operating mode, the interrupt logic, and the status register which provides operational feedback to the CPU.

Timing

The timing section contains the counters and decoding logic necessary to generate the monitor timing outputs and to control the display format. These timing parameters are selected by programming of the initialization registers.

Display Control

The display control section generates linear addressing for up to 16K bytes of display memory. Internal comparators limit the portion of the memory which is displayed to programmed values. Additional functions performed in this section include cursor positioning, storage of light pin 'hit' location, and address comparisons required for generation of timing signals and the split screen interrupt.

Buffer Control

The buffer control section generates three signals which control the transfer of data between the CPU and the display buffer memory. Four system configurations requiring four different 'handshaking' schemes are supported. These are described below.

SYSTEM CONFIGURATIONS

Figure 1 illustrates the block diagram of a typical display terminal using the Signetics 2670, 2671, 2672, and 2673/2677 CRT terminal devices. In this system, the CPU examines inputs from the data communications line and the keyboard and places the data to be displayed in the display buffer memory. This buffer is typically a RAM which holds the data for a single or multiple screenload (page) or for a single character row.

The PVTC supports four common system configurations of display buffer memory: the independent, transparent, shared, and row buffer modes. The first three modes utilize a single or multiple page RAM and differ primarily in the means used to transfer display data between the RAM and the CPU. The row buffer mode makes use of a single row buffer (which can be a shift register or a small RAM) that is updated in real time to contain the appropriate display data.

The user programs bits 0 and 1 of IR0 to select the mode best suited for the system environment. The CNTRL1 – 3 outputs perform different functions for each mode and are named accordingly in the description of each mode.

Independent Mode

The CPU to RAM interface configuration for this mode is illustrated in figure 2. Transfer of data between the CPU and display memory is accomplished via a bidirectional latched port and is controlled by the signals read data buffer ($\overline{RDB}$), write data buffer ($\overline{WDB}$), and

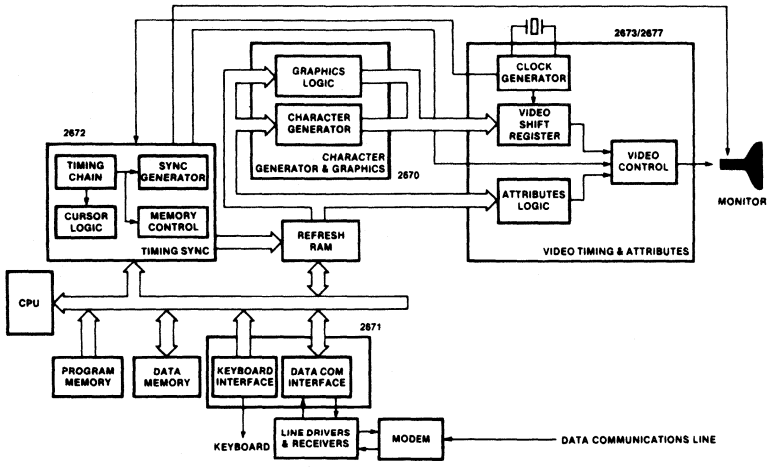


Figure 1. CRT Terminal Block Diagram

buffer chip enable ($\overline{BCE}$). This mode provides a non-contention type of operation that does not require address multiplexers. The CPU does not address the memory directly – the read or write operation is performed at the address contained in the cursor address register or the pointer address register as specified by the CPU. The PVTC enacts the data transfers during blanking intervals in order to prevent visual disturbances of the displayed data.

For a data buffer write command, the $\overline{WDB}$ signal will go active on the rising edge of character clock ($\overline{CCLK}$) and will remain so until the next $\overline{CCLK}$ rising edge. $\overline{BCE}$ is always in the active state except before and after a $\overline{WDB}$ command. When a write has been executed, $\overline{BCE}$ becomes inactive on the falling edge of $\overline{CCLK}$. When the write occurs ($\overline{WDB}$ active) on the next rising edge, $\overline{BCE}$ also becomes active. $\overline{BCE}$ and $\overline{WDB}$ both become inactive on the rising edge of $\overline{CCLK}$. $\overline{BCE}$ will then return to the active state on the next falling edge if there is no other write command to be executed.

The CPU manages the data transfers by supplying commands to the PVTC. The commands used are:

1. Read/Write at pointer address.
2. Read/Write at cursor address (with optional increment of address).
3. Write from cursor address to pointer address.

The operational sequence for a write operation is:

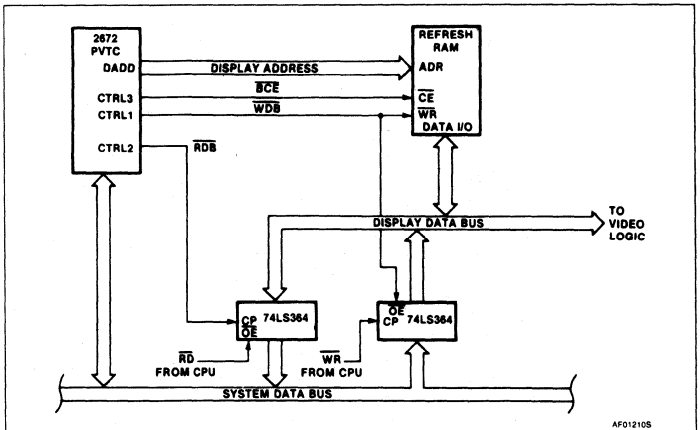
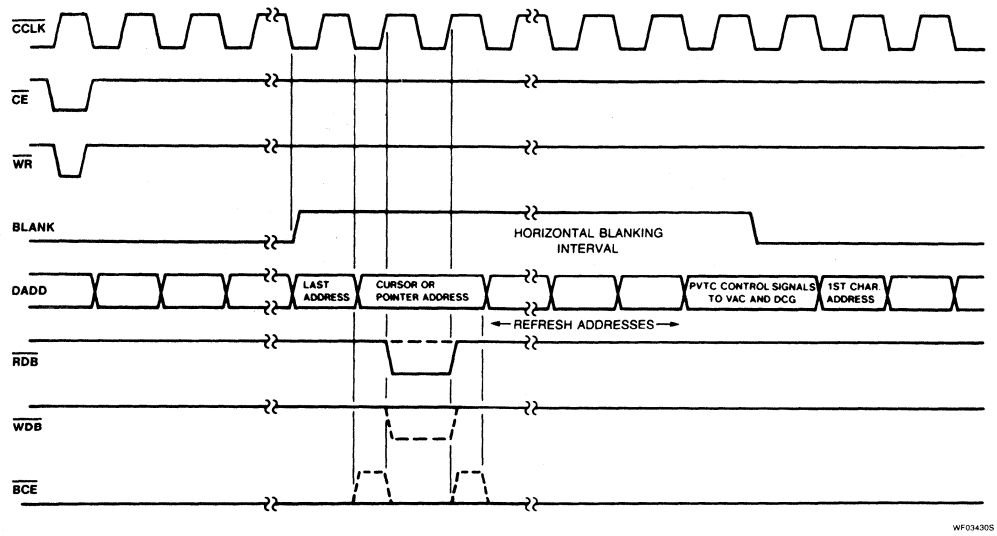


Figure 2. Independent Buffer Mode Configuration

1. CPU checks RDFLG status bit to assure that any previous operation has been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes address into cursor or pointer registers.
4. CPU issues 'write at cursor with/without increment' or 'write at pointer' command.
5. PVTC generates control signals and outputs specified address to perform requested operation. Data is copied from the interface latch into the memory.
6. PVTC sets RDFLG status to indicate that the write is completed.

Similarly, a read operation proceeds as follows:

1. Steps 1 and 3 as above.
2. CPU issues 'read at cursor with/without increment' or 'read at pointer' command.
3. PVTC generates control signals and outputs specified address to perform requested operation. Data is copied from memory to the interface latch and PVTC sets RDFLG status to indicate that the read is completed.



NOTE:
Write waveforms shown in dotted lines.

Figure 3. Read/Write at Cursor/Pointer Command Timing (Command Received During Active Display Window)

4. CPU checks RDFLG status to see if operation is completed.
5. CPU reads data from interface latch.

Loading the same data into a block of display memory is accomplished via the 'write from cursor to pointer' command:

1. CPU checks RDFLG status bit to assure that any previous operation has been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes beginning address of memory block into cursor address register and ending address of block into pointer address register.
4. CPU issues 'write from cursor to pointer' command.
5. PVTC generates control signals and outputs block addresses to copy data from the interface latch into the specified block of memory.
6. PVTC sets RDFLG status to indicate that the block write is completed.

Similary sequences can be implemented on an interrupt driven basis using the READY interrupt output to advise the CPU that a previously requested command has been completed.

Two timing sequences are possible for the 'read/write at cursor/pointer' commands. If the command is given during the active display window (defined as first scan line of the first character row to the last scan line of the last character row), the operation takes place during the next horizontal blanking interval, as illustrated in figure 3. If the command is given during the vertical blanking interval, or while the display has been commanded blanked, the operation takes place immediately. In the latter case, the execution time for the command is approximately one microsecond plus six (6) character clocks (see figure 4).

Timing for the 'write from cursor to pointer' operation is shown in figure 5. The BLANK output is asserted automatically and remains asserted until the horizontal retrace interval following completion of the command. The memory is filled at a rate of one location per two character times, plus a small amount of overhead.

Shared and Transparent Buffer Modes

In these modes the display buffer RAM is a part of the CPU memory domain and is addressed directly by the CPU. Both modes use the same hardware configuration with the CPU accessing the display buffer via three-state drivers (see figure 6). The processor bus request ($\overline{PBREQ}$) control signal informs the PVTC that the CPU is requesting access to the display buffer. In response to this request, the PVTC raises bus acknowledge ($\overline{BACK}$) until its bus external ($\overline{BEXT}$) output has freed the display address and data busses for CPU access. $\overline{BACK}$, which can be used as a 'hold' input to the CPU, is then lowered to indicate that the CPU can access the buffer.

In transparent mode, the PVTC delays the granting of the buffer to the CPU until a vertical or horizontal blanking interval, thereby causing minimum disturbance of the display. In shared mode, the PVTC will blank the display and grant immediate access to the CPU. Timing for these modes is illustrated in figures 7, 8, and 9.

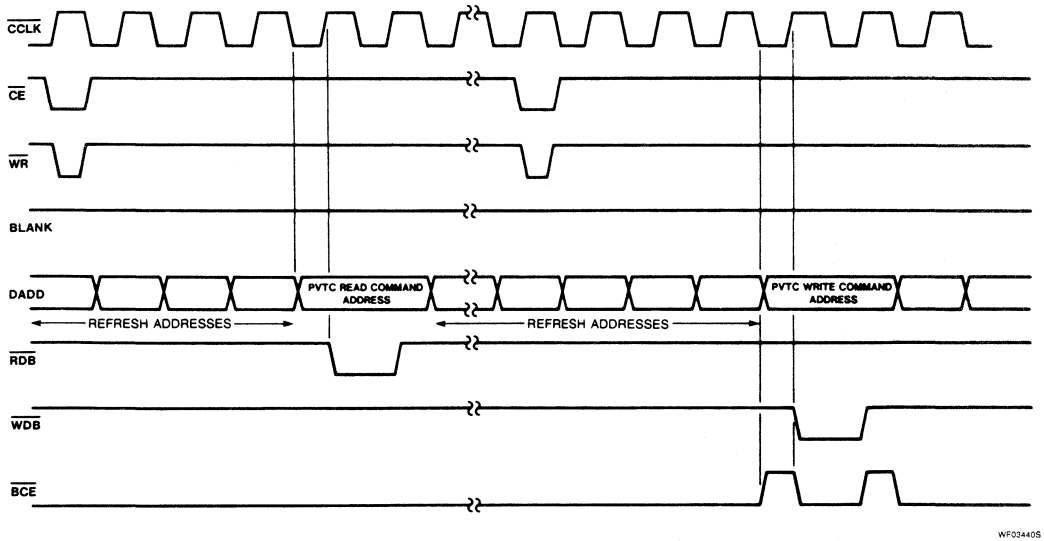


Figure 4. Read/Write at Cursor/Pointer Command Timing (Command Received While Display is Blanked)

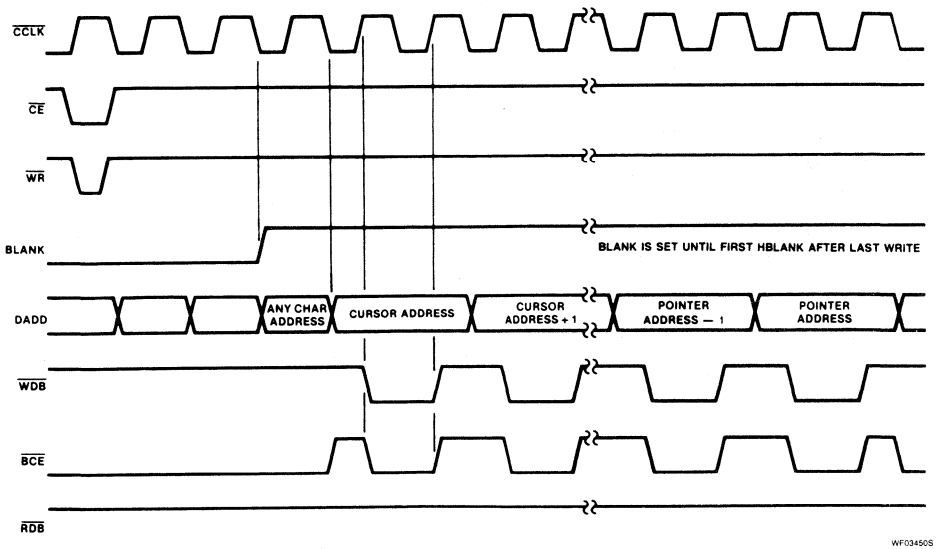


Figure 5. Write from Cursor to Pointer Command Timing

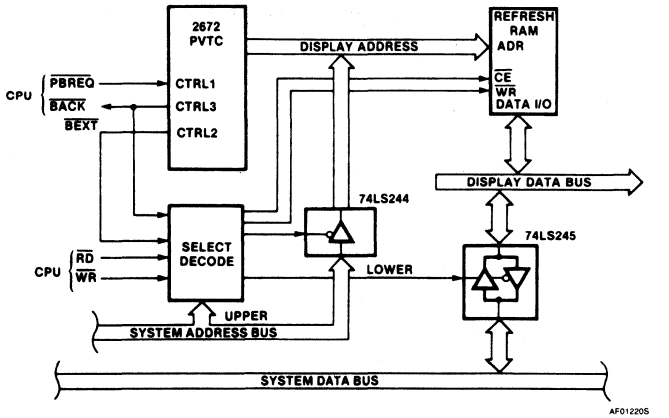
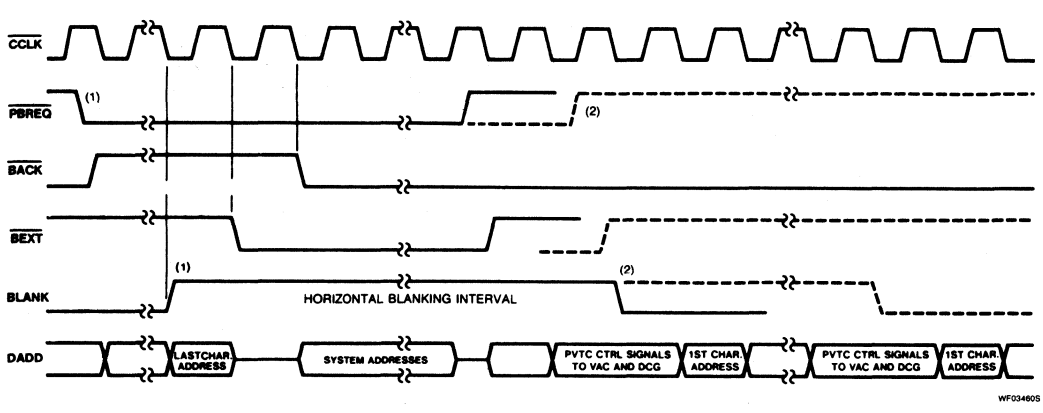
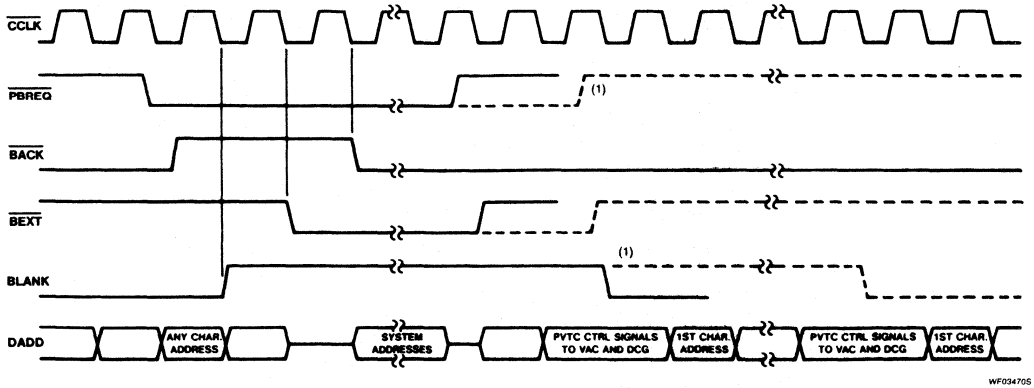


Figure 6. PVTC Shared or Transparent Buffer Modes



NOTES:
1. $\overline{\text{PBREQ}}$ must be asserted prior to the rising edge of BLANK in order for sequence to begin during that blanking period.
2. If $\overline{\text{PBREQ}}$ is negated after the next to last $\overline{\text{CCLK}}$ of the horizontal blanking interval, the next scan line will also be blanked.

Figure 7. Transparent Buffer Mode Timing



NOTE:
1. If $\overline{PBREQ}$ is negated after the next to last $\overline{CCLK}$ of the horizontal blanking interval, the next scan line will also be blanked.

Figure 8. Shared Buffer Mode Timing

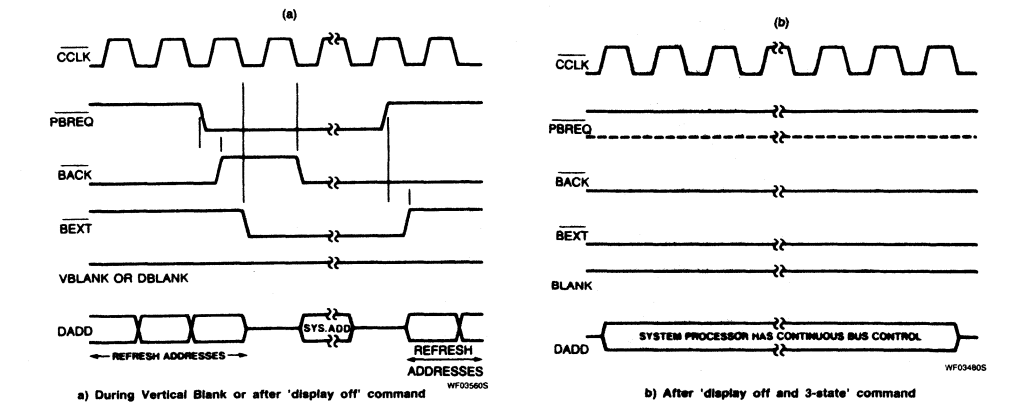


Figure 9. Shared and Transparent Mode Timing

Row Buffer Mode

Figures 10 and 11 show the timing and a typical hardware implementation for the row buffer mode. During the first scan line (line 0) of each character row, the PVTC halts the CPU and DMA's the next row of character data from the system memory to the row buffer memory. The PVTC then releases the CPU and displays the row buffer data for the programmed number of scan lines. The bus request control ($\overline{BREQ}$) signal informs the CPU that character addresses and the memory bus control (MBC) signal will start at the next falling edge of BLANK. The CPU must release the address and data busses before this time to prevent bus contention. After the row of character data is transferred to the row buffer RAM, $\overline{BREQ}$ returns high to grant memory control back to the CPU.

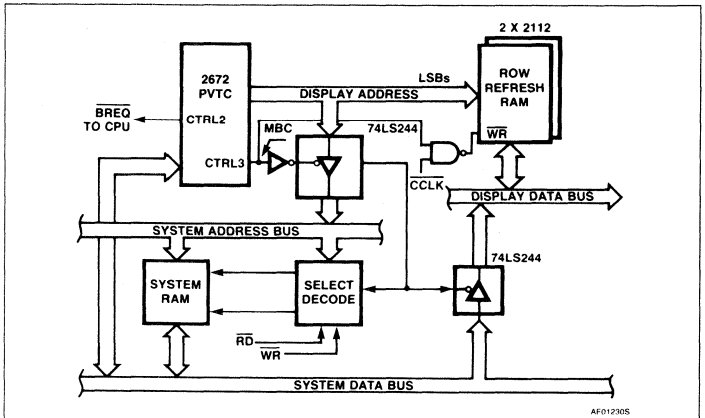


Figure 10. Row Buffer Mode Configuration

OPERATION

After power is applied, the PVTC will be in an inactive state. Two consecutive 'master reset' commands are necessary to release this circuitry and ready the PVTC for operation. Two register groups exist within the PVTC: the initialization registers and the display control registers. The initialization registers select the system configuration, monitor timing, cursor shape, display memory domain, and screen format. These are loaded first and normally require no modification except for certain special visual effects. The display control registers specify the memory address of the base character (upper left corner of screen), the cursor position, and the pointer address for independent memory access mode. These usually require modification during operation.

After initial loading of the two register groups, the PVTC is ready to control the monitor screen. Prior to executing the PVTC commands which turn on the display and cursor, the user should load the display memory with the first data to be displayed. During operation, the PVTC will sequentially address the display memory within the limits programmed into its registers. The memory outputs character codes to the system character and graphics generation logic, where they are converted to the serial video stream necessary to display the data on the CRT. The user effects changes to the display by modifying the contents of the display memory, the PVTC display control and command registers, and the initialization registers, if required. Interrupts and status conditions generated by the PVTC supply the 'handshaking' information necessary for the CPU to effect the display changes in the proper time frame.

Initialization Registers

There are 11 initialization registers (IR0 – IR10) which are accessed sequentially via a single address. The PVTC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR10, the split screen register) is accessed. The pointer then continues to point to the split screen register. Upon power-up or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command. These registers are write only and are used to specify parameters such as the system configuration, display format, cursor shape, and monitor timing. Register formats are shown in table 2.

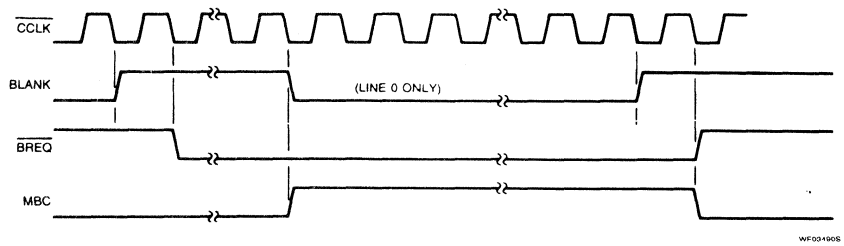


Figure 11. Row Buffer Mode Timing

Table 2. INITIALIZATION REGISTER BIT FORMATS

IR0

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	SCAN LINES PER CHARACTER ROW				SYNC SELECT	BUFFER MODE SELECT	
	NON-INTERLACED		INTERLACED				
NOT USED	0000 = 1 LINE 0001 = 2 LINES 0010 = 3 LINES • • 1110 = 15 LINES 1111 = 16 LINES		0000 = UNDEFINDED 0001 = 5 LINES 0010 = 7 LINES • • 1110 = 31 LINES 1111 = UNDEFINED		0 = VSYNC 1 = CSYNC	00 = INDEPENDENT 01 = TRANSPARENT 10 = SHARED 11 = ROW	

IR1

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
INTERLACE ENABLE	EQUALIZING CONSTANT						
0 = NON-INT 1 = INTER.	00000000 = 1 CCLK 00000001 = 2 CCLK • • 11111110 = 127 CCLK 11111111 = 128CCLK		CALCULATED FROM: $EC = 0.5(H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}) - 2(H_{SYNC})$				

IR2

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
	HORIZONTAL SYNC WIDTH				HORIZONTAL BACK PORCH		
NOT USED	0000 = 2 CCLK 0001 = 4 CCLK • • 1110 = 30 CCLK 1111 = 32 CCLK				000 = 1 CCLK 001 = 5 CCLK • • 110 = 25 CCLK 111 = 29 CCLK		

IR3

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
VERTICAL FRONT PORCH			VERTICAL BACK PORCH				
000 = 4 SCAN LINES 001 = 8 SCAN LINES • • 110 = 28 SCAN LINES 111 = 32 SCAN LINES			00000 = 4 SCAN LINES 00001 = 6 SCAN LINES • • 11110 = 64 SCAN LINES 11111 = 66 SCAN LINES				

IR4

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CHARACTER BLINK RATE	ACTIVE CHARACTER ROWS PER SCREEN (NOTE 1)						
0 = 1/16 VSYNC 1 = 1/32 VSYNC	00000000 = 1 ROW 00000001 = 2 ROWS • • 11111110 = 127 ROWS 11111111 = 128 ROWS						

NOTE:
1. In interlace mode with odd total character rows per screen the last character row will be the programmed scan lines per character row minus one.

Table 2. INITIALIZATION REGISTER BIT FORMATS (Continued)

IR5

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ACTIVE CHARACTERS PER ROW							
00000010 = 3 CHARACTERS 00000011 = 4 CHARACTERS • • 11111110 = 255 CHARACTERS 11111111 = 256 CHARACTERS							

IR6

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
FIRST LINE OF CURSOR				LAST LINE OF CURSOR			
0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15				0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR7

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LIGHT PEN LINE		CURSOR BLINK	DOUBLE HEIGHT CHAR.	UNDERLINE POSITION			
00 = SCAN LINE 3 01 = SCAN LINE 5 10 = SCAN LINE 7 11 = SCAN LINE 9		0= NO 1= YES	0= NO 1= YES	0000 = SCAN LINE 0 0001 = SCAN LINE1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR8

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER FIRST ADDRESS LSB'S							
H'000' = 0 H'001' = 1 • • H'FFE' = 4,094 H'FFF' = 4,095				NOTE: MSB'S ARE IN IR9[3:0]			

IR9

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER LAST ADDRESS				DISPLAY BUFFER FIRST ADDRESS MSB'S			
0000 = 1,023 0001 = 2,047 • • 1110 = 15,359 1111 = 16,383				SEE IR8			

IR10

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CURSOR BLINK RATE	SPLIT SCREEN INTERRUPT ROW						
0 = 1/16 VSYNC 1 = 1/32 VSYNC	00000000 = ROW 0 00000001 = ROW 1 • • 11111110 = ROW 126 11111111 = ROW 127						

IR0[6:3] – Scan Lines per Character Row
Both interlaced and non-interlaced scanning are supported by the PVTC. For interlaced mode, two different formats can be implemented, depending on the interconnection between the PVTC and the character generator (see IR1[7]). This field defines the number of scan lines used to compose a character row for each technique. As scanning occurs, the scan line count is output on the LA0 – LA3 and LI pins.

IR0[2] – VS/CS Enable
This bit selects either vertical sync pulses or composite sync pulses on the VSYNC/CSYNC output (pin 18). The composite sync waveform conforms to EIA RS170 standards, with the vertical interval composed of six equalizing pulses, six vertical sync pulses, and six more equalizing pulses.

IR0[1:0] – Buffer Mode Select
Four buffer memory modes may be selectively enabled to accommodate the desired system configuration. See System Configurations.

IR1[7] – Interface Enable
Specifies interlaced or noninterlaced timing operation. Two modes of interlaced operation are available, depending on whether LA0 – LA3 or LI, LA0 – LA 2 are used as the line address for the character generator. The resulting displays are shown in figure 12.

For 'interlaced sync' operation, the same information is displayed in both odd and even fields, resulting in enhanced readability. The PVTC outputs successive line numbers in ascending order on the LA0 – LA3 lines, one per scan line for each field.

The 'interlaced sync and video' format doubles the character density on the screen. The PVTC outputs successive line numbers in ascending order on the LI, LA0 – LA2 lines, one per scan line for each field, but alternates beginning the count with even and odd line numbers. In the interlaced sync and video mode, the number of scan lines per character row is always odd. Assume that the first character row is row 0 (even). When in the odd field, the scan line numbers being displayed are even for even character rows and odd for odd character rows. When in the even field, the scan line numbers being displayed are odd for even character rows and even for odd character rows (see figure 12c). This provides balanced beam currents in the odd and even fields, thus minimizing character variations due to different loading of the CRT anode supply between fields.

IR1[6:0] – Equalizing Constant
This field indirectly defines the horizontal front porch and is used internally to generate the equalizing pulses for the RS170 compatible CSYNC. The value for this field is the total number of character clocks ($\overline{CCLK}$) during a horizontal line period divided by two, minus two times the number of character clocks in the horizontal sync pulse:

$$EC = \frac{H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}}{2} - 2(H_{SYNC})$$

The definition of the individual parameters is illustrated in figure 13. The minimum value of H_{FP} is two character clocks.

Note that when using the 2673/2677 VAC, the blank pulse is delayed three CCLKs relative to the HSYNC pulse. Because of this delay, the actual HFP and HBP values will be different from the values programmed into the PVTC. The actual HFP will be decreased by 3 character clocks. The actual HBP will be increased by 3 character clocks.

IR2[6:3] – Horizontal Sync Pulse Width
This field specifies the width of the HSYNC pulse in $\overline{CCLK}$ periods.

IR2[2:0] – Horizontal Back Porch
This field defines the number of $\overline{CCLK}$s between the trailing edge of HSYNC and the trailing edge of BLANK.

IR3[7:5] – Vertical Front Porch
Programs the number of scan line periods between the rising edges of BLANK and VSYNC during a vertical retrace interval. The width of the VSYNC pulse is fixed at three scan lines.

IR3[4:0] – Vertical Back Porch
This field determines the number of scan line periods between the falling edges of the VSYNC and BLANK outputs.

IR4[7] – Character Blink Rate
Specifies the frequency for the character blink attribute timing. The blink rate can be specified as 1/16 or 1/32 of the vertical field rate. The timing signal has a duty cycle of 75% and is multiplexed onto the DADD11/BLINK output at the falling edge of each BLANK.

IR4[6:0] – Character Rows Per Screen
This field defines the number of character rows to be displayed. This value multiplied by the scan lines per character row, plus the vertical front and back porch values, and the vertical sync pulse width (three scan lines) is the vertical scan period in scan lines.

IR5[7:0] – Active Characters Per Row
This field determines the number of characters to be displayed on each row of the CRT screen. The sum of this value, the horizontal front porch, the horizontal sync width, and the horizontal back porch is the horizontal scan period in $\overline{CCLK}$s.

IR6[7:4], IR6[3:0] – First and Last Scan Line of Cursor
These two fields specify the height and position of the cursor on the character block. The 'first' line is the topmost line when scanning from the top to the bottom of the screen. The value of the first line of cursor must be less than the last line of cursor value.

IR7[7:6] – Light Pen Line Position
This field defines which of four scan lines of the character row will be used for the light pen strike – through attribute by the 2673/2677 VAC. The timing signal is multiplexed onto the DADD9/LPL output during the falling edge of BLANK.

IR7[5] – Cursor Blink Enable
This bit controls whether or not the cursor output pin will be blinked at the selected rate (IR10[7]). The blink duty cycle for the cursor is 50%.

IR7[4] – Double Height Character Row Enable
If enabled, the scan line count will be repeated twice in succession, causing the height of the character row to double. This bit can be changed at any time but will only become effective at the beginning of the character row following the time it is changed. This allows selected character rows to be of double height. The split screen interrupt can be used to notify the CPU when to effectuate changes to this bit. For each double height row which replaces a normal row, one row count should be subtracted from the 'character rows per screen' field (IR4) to maintain the same total number of scan lines per field.

IR7[3:0] – Underline Position
This field defines which scan line of the character row will be used for the underline attribute by the 2673/2677 VAC. The timing signal is multiplexed onto the DADD10/UL output during the falling edge of BLANK.

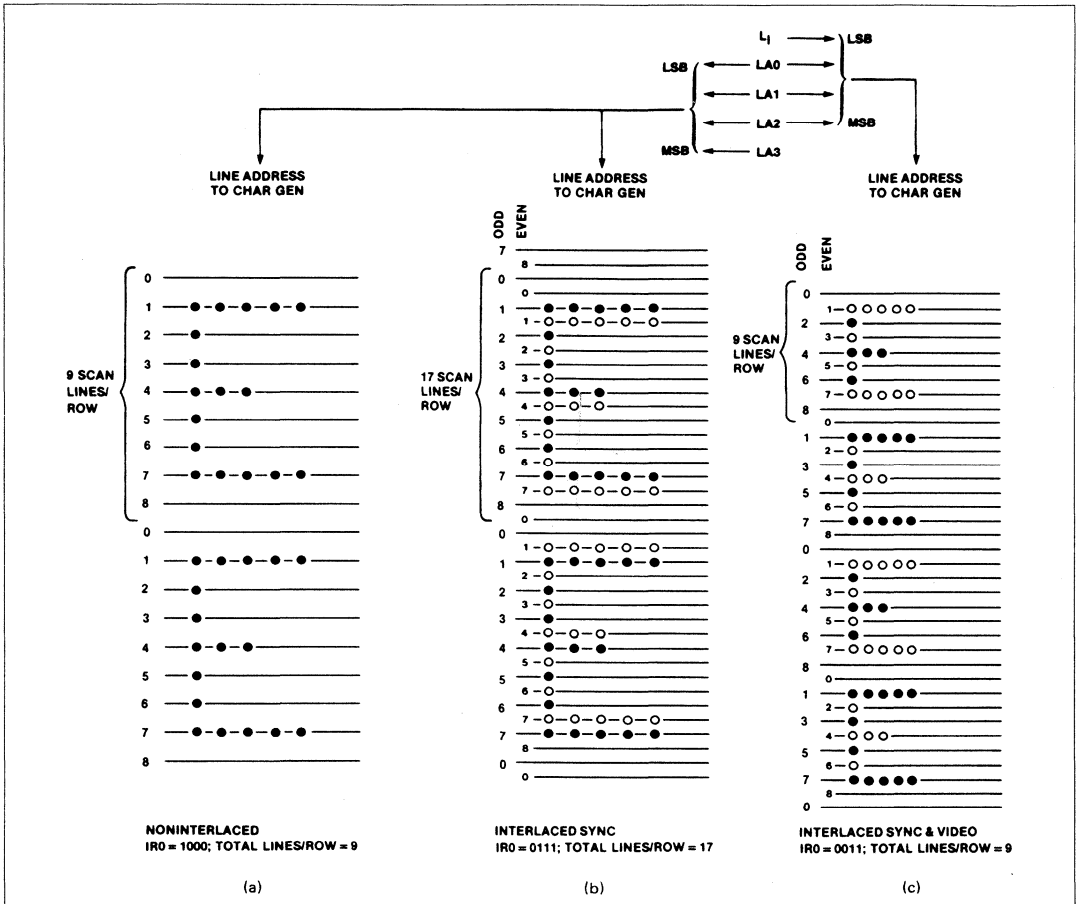


Figure 12. Interlaced Display Modes

IR9[3:0], IR8[7:0] – Display Buffer First Address

IR9[7:4] – Display Buffer Last Address

These two fields define the area within the buffer memory where the display data will reside. When the data at the 'display buffer last address' is displayed, the PVTC will wraparound and obtain the data to be displayed at the next screen position from the 'display buffer first address'. If 'last address' is the end of a character row and a new screen start address has been loaded into the screen start register, or if 'last address' is the last character position of the screen, the next data is obtained from the address contained in the screen start register.

Note that there is no restriction in displaying data from other areas of the addressable memory. Normally, the area between these two bounds is used for data which can be overwritten (e.g., as a result of scrolling), while data that is not to be overwritten would be contained outside these bounds and accessed by means of the split screen interrupt feature of the PVTC.

IR10[7] – Cursor Blink Rate

The cursor blink rate can be specified at $^1/_{16}$ or $^1/_{32}$ of the vertical scan frequency. Blink is effective only if blink is enabled by IR7[5].

IR10[6:0] – Split Screen Interrupt

The split screen interrupt can be used to provide special screen effects such as a row of double height characters or to change the normal addressing sequence of the display memory. The contents of this field is compared, in real time, to the current character row number. Upon a match, the PVTC sets the split screen status bit, and issues an interrupt request if so programmed. The status change/interrupt request is made at the beginning of scan line zero of the split screen character row.

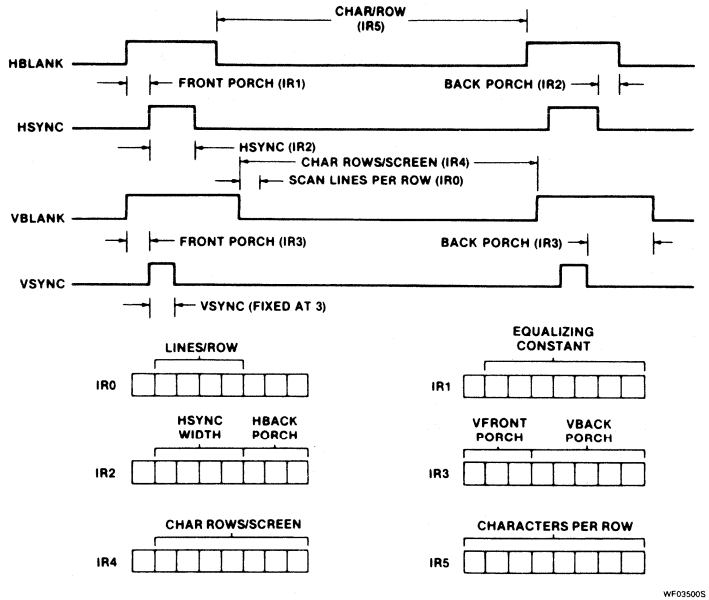


Figure 13. Horizontal and Vertical Timing

Timing Considerations

Normally, the contents of the initialization registers are not changed during operation. However, this may be necessary to implement special display features such as multiple cursors, smooth scrolling, horizontal scrolling, and double height character rows. Table 3 describes timing details for these registers which should be considered when implementing these features.

Table 3. TIMING CONSIDERATIONS

PARAMETER	TIMING CONSIDERATIONS
First line of cursor Last line of cursor Light pen line Underline	These parameters must be established at a minimum of two character times prior to their occurrence.
Double height characters	Set/reset during the character row prior to the affected row.
Cursor blink Cursor blink rate Character blank rate	New values become effective within one field after values are changed
Split screen interrupt row	Change anytime prior to line zero of desired row
Character rows per screen	Change only during vertical blanking period
Vertical front porch	Change prior to first line of V_{FP}
Vertical back porch	Change prior to fourth line after V_{SYNC}
Screen start register	Change prior to the horizontal blanking interval of the last line of character row prior to the affected row.

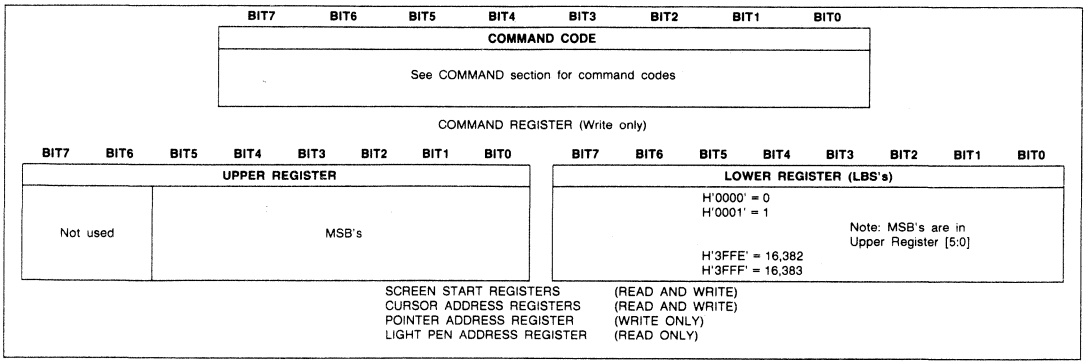
Display Control Registers

There are nine registers in this group, each with an individual address. Their formats are illustrated in table 4. The command register is used to invoke one of 16 possible PVTC commands as described in the COMMANDS section of this data sheet. The remaining registers in the group store address values which specify the cursor and buffer pointer locations, the location of the first character to be displayed on the screen, and the location of a light pen 'hit'. With the exception of the light pen register, the user initializes these registers after powering on the system and changes their values to control the data which is displayed.

Screen Start Registers

The screen start registers contain the address of the first character of the first row (upper left corner of the active display). At the beginning of the first scan line of the first row, this address is transferred to the row start register (RSR) and into the memory address counter (MAC). The counter is then advanced sequentially at the character rate the number of times programmed into the active characters per row register (IR5), thus reaching the address of the last character of the row plus one. At the beginning of each subsequent scan line of the first row, the MAC is reloaded from the RSR and the above sequence is repeated. At the end of the last scan line of the first row, the contents of the MAC are loaded into the RSR to serve as the starting memory address for the second character

Table 4. DISPLAY CONTROL REGISTER FORMATS



row. This process is repeated for the programmed number of rows per screen. Thus, the data in the display memory is displayed sequentially starting from the address contained in the screen start register. After the ensuing vertical retrace interval the contents of the screen registers are reloaded into the RSR and MAC and the process is repeated.

The sequential operation described above will be modified upon the occurrence of either of two events. First, if during the incrementing of the memory address counter the 'display buffer last address' (IR9[7:4]) is reached, the MAC will be loaded from the 'display buffer first address' register (IR9[3:0], IR8[7:0]), at the next character clock. Sequential operation will then resume starting from this address. This wraparound operation allows portions of the display buffer to be used for purposes other than storage of displayable data and is completely automatic without any CPU intervention (see figure 14a).

Second, the sequential row to row addressing can also be modified under CPU control. If the contents of the screen start register (upper, lower, or both) are changed during any character row (say row 'n'), the starting address of the next character row (row 'n + 1') will be the new value of the screen start register and addressing will continue sequentially from there. This allows features such as split screen operation, partial scroll, or status line display to be implemented. The split screen interrupt feature of the PVTC is useful in controlling this type of operation. Note that in order to obtain the correct screen display, the screen start register must be reloaded with the original value prior to the end of the vertical retrace. See figure 14b.

Refresh Addressing

During vertical blanking the address counter operation is modified by stopping the automatic load of the contents of the RSR into the counter, thereby allowing the address outputs to free-run. This allows dynamic memory refresh to occur during the vertical retrace interval. The refresh addressing starts at the last address displayed on the screen and increments by one for each character clock during the retrace interval. If the display buffer last address is encountered, wraparound will occur and refreshing will continue from the display buffer first address.

Cursor Address Registers

The contents of these registers define the buffer memory address of the cursor. If enabled, the cursor output will be asserted when the memory address counter (MAC) matches the value of the cursor address registers. The cursor address registers may be read or written by the CPU or incremented via the 'increment cursor address' command. In independent buffer mode, these registers define a buffer memory address for PVTC controlled access in response to 'read/write at cursor with/without increment' commands, or the first address to be used in executing the 'write from cursor to pointer' command.

Display Pointer Address Registers

These registers define a buffer memory address for PVTC controlled accesses in response to 'read/write at pointer' commands. They also define the last buffer memory address to be written for the 'write from cursor to pointer' command.

Light Pen Address Registers

If the light pen input is enabled, these registers are used to store the current character address upon receipt of a light pen strobe input. Several sources of delay between the display of a character upon the screen and the receipt of a light pen hit can be expected to exist in a system environment. These delays include address pipelining in the character generation circuits, delays in the video generation circuits, and delays in the light detection circuitry itself. These delays cause the value stored in the light pen register to differ from the actual address of the character at which the light pen hit actually was detected. Software must be used to correct this condition.

Interrupt/Status Registers

The interrupt and status registers provide information to the CPU to allow it to interact with the PVTC to effect desired changes to implement various display operations. The interrupt register provides information on five possible interrupting conditions, as shown in table 5. These conditions may be selectively enabled or disabled (masked) from causing interrupts by certain PVTC commands. An interrupt condition which is enabled (mask bit equal to one) will cause the $\overline{INTR}$ output to be asserted and will cause the corresponding bit in the interrupt register to be set upon occurrence of the interrupting condition. An interrupt condition which is disabled (mask bit equal to zero) has no effect on either the $\overline{INTR}$ output or the interrupt register.

The status register provides six bits of status information: the five possible interrupting conditions plus the NOT BUSY bit. For this register, however, the contents are not effected by the state of the mask bits.

Descriptions of each interrupt/status register bit follow. Unless otherwise indicated, a bit, once set, will remain set until reset by the CPU by issuing a 'reset interrupt/status bits' command. The bits are also reset by a 'master reset' command and upon power-up. This bit is set to a one upon a master reset.

SR[5] – RDFLG

This bit is present in the status register only. A zero indicates that the PVTC is currently executing the previously issued command. A one indicates that the PVTC is ready to accept a new command.

Table 5. INTERRUPT AND STATUS REGISTER FORMAT

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Not used always read as 0		RDFLG	VBLANK	LINE ZERO	SPLIT SCREEN	READY	LIGHT PEN
		0 = Busy 1 = Ready *	0 = No 1 = Yes	0 = No 1 = Yes	0 = No 1 = Yes	0 = Busy 1 = Ready	0 = No 1 = Yes

NOTE:
*Status register only. Always 0 when reading interrupt register.

I/SR[4] – VBLANK
Indicates the beginning of a vertical blanking interval. Is set to a one at the beginning of the first scan line of the vertical front porch.

I/SR[3] – Line Zero
Is set to a one at the beginning of the first scan line (line 0) of each active character row.

I/SR[2] – Split Screen
This bit is set when a match occurs between the current character row number and the value contained in the split screen interrupts register, IR10[6:0]. The equality condition is only checked at the beginning of line zero of each character row. This bit is reset when either of the screen start registers is loaded by the CPU.

I/SR[1] – Ready
Certain PVTC commands affect the display and may require the PVTC to wait for a blanking interval before enacting the command. This bit is set to one when execution of the command has been completed. No command should be invoked until the prior command is completed. This bit is set to a zero upon a master reset.

I/SR[0] – Light Pen
A one indicates that a light pen hit has occurred and that the contents of the light pen register have been updated. This bit will be reset when either of the light pen registers is read.

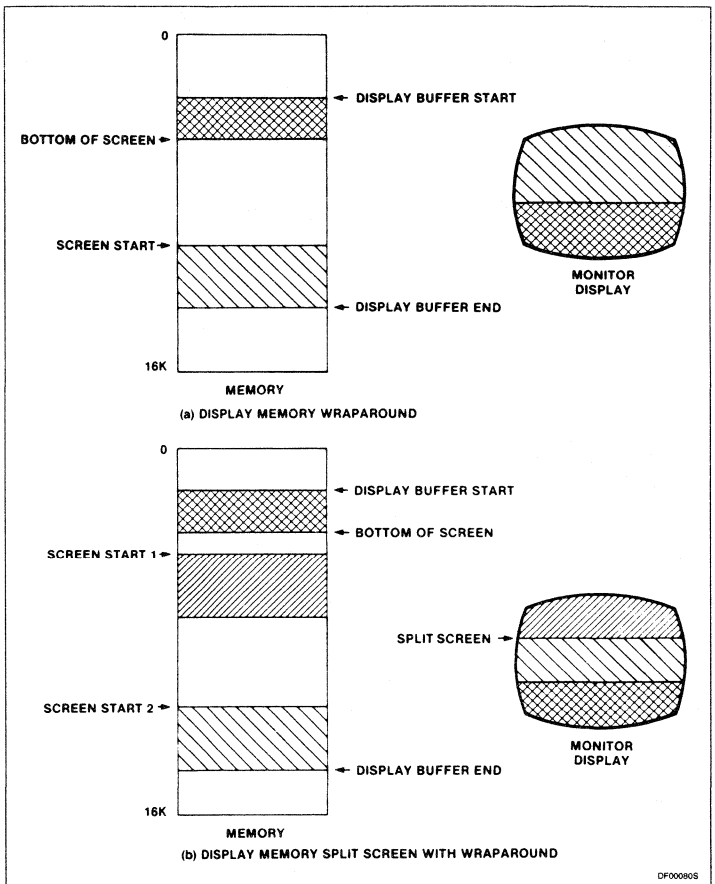


Figure 14. Display Addressing Operation

COMMANDS

The PVTC commands are divided into two classes: the instantaneous commands, which are executed immediately after they are invoked, and the delayed commands which may need to wait for a blanking interval prior to their execution. Command formats are shown in table 6. The commands are asserted by performing a write operation to the command register with the appropriate bit pattern as the data byte.

Instantaneous Commands

The instantaneous commands are executed immediately after the trailing edge of the $\overline{WR}$ pulse during which the command is issued. These commands do not affect the state of the RDFLG or READY interrupt/status bits.

However, a command should not be invoked if the RDFLG bit is low.

Master Reset
This command initializes the PVTC and may be invoked at any time to return the PVTC to its initial state. Upon power-up, two successive master reset commands must be applied to release the PVTC's internal power on circuits. In transparent and shared buffer modes, the CNTRL1 input must be high when the command is issued. The command causes the following:

1. VSYNC and HSYNC are driven low for the duration of RESET and BLANK goes high. BLANK remains high until a 'display on' command is received.
2. The interrupt and status bits and masks are set to zero, except for the RDFLG flag which is set to a one.
3. The transparent mode, cursor off, display off, and light pen disable states are set.
4. The initialization register pointer is set to address IR0.

Load IR Address
This command is used to preset the initialization register pointer with the value 'V' defined by D3 – D0. Allowable values are 0 to 10.

Enable Light Pen
After invoking this command, receipt of a light pen strobe input will cause the light pen register to be loaded with the current buffer memory address and the corresponding interrupt and status flag to be set. Once loaded, further loads are inhibited until either one of the light pen registers are read or a reset function is performed.

Disable Light Pen
Light pen hits will not be recognized.

Display Off
Asserts the BLANK output. The DADD0 through DADD13 display address bus outputs may be optionally placed in the three-state condition by setting bit 2 to a '1' when invoking the command.

Display On
Restores normal blanking operation either at the beginning of the next field (bit 2 = 1) or at the beginning of the next scan line (bit 2 = 0). Also returns the DADD0-DADD13 drivers to their active state.

Cursor Off
Disables cursor operation. Cursor output is placed in the low state.

Cursor On
Enables normal cursor operation.

Reset Interrupt/Status Bits
This command resets the designated bits in the interrupt and status registers. The bit positions correspond to the bit positions in the registers:

Bit 0 – Light pen

Bit 1 – Ready

Bit 2 – Split screen

Bit 3 – Line zero

Bit 4 – Vertical blank

Disable Interrupts
Sets the interrupt mask to zeros for the designated conditions, thus disabling these conditions from asserting the $\overline{INTR}$ output. Bit position correspondence is as above.

Enable Interrupts
Resets the selected interrupt and status register bits and writes the associated interrupt mask bits to a one. This enables the corresponding conditions to assert the $\overline{INTR}$ output. Bit position correspondence is as above.

Delayed Commands

This group of commands is utilized for the independent buffer mode of operation, although the 'increment cursor' command can also be used in other modes. With the exception of the 'write from cursor to pointer' and 'increment cursor' commands, all the commands of this type will be executed immediately or will be delayed depending on when the command is invoked. If invoked during the active screen time, the command is executed at the next horizontal blanking interval. If invoked during a vertical retrace interval or a 'display off' state, the command is executed immediately.

The 'increment cursor' and 'write from cursor to pointer' commands are executed immediately after they are issued. 'Increment cursor' requires approximately three $\overline{CCLK}$ periods for completion. 'Write from cursor to pointer' asserts the BLANK output during its execution. BLANK will not be released until the beginning of the horizontal blanking interval following the last write operation. This will allow more than one 'write from cursor to pointer' command to be executed during one frame and will blank the screen for the time required to execute the command.

In all cases, the PVTC will assert the READY/RDFLG status to signify completion of the command. No other commands should be given until the current command is completed. Therefore, the READY interrupt or RDYFLG status flag should be used for handshaking control between the PVTC and CPU when using these commands.

Read/Write at Pointer
Transfers data between the display buffer the bus interface latch using the address contained in the pointer register.

Table 6. PVTC COMMAND FORMATS

D7	D6	D5	D4	D3	D2	D1	D0	Hex	COMMAND
Instantaneous Commands:									
0	0	0	0	0	0	0	0		Master reset
0	0	0	1	V	V	V	V		Load IR pointer with value V (V = 0 to 10)
0	0	1	d	d	d	1	0[1]		Disable light pen
0	0	1	d	d	d	1	1[2]		Enable light pen
0	0	1	d	1	N	d	0[1]		Display off. Float DADD bus if N = 1
0	0	1	d	1	N	d	1[2]		Display on: Next field (N = 1) or scan line (N = 0)
0	0	1	1	d	d	d	0[1]		Cursor off
0	0	1	1	d	d	d	1[2]		Cursor on
0	1	0	N	N	N	N	N		Reset interrupt/status: Bit reset where N = 1
1	0	0	N	N	N	N	N		Disable interrupt: Disable where N = 1
0	1	1	N	N	N	N	N		Enable interrupt: Enables interrupts and resets the corresponding interrupt/status bits where N = 1
			V B	L Z	S S	R D	L P		
Delayed Commands:								**Hex**	
1	0	1	0	0	1	0	0	A4	Read at pointer address
1	0	1	0	0	0	1	0	A2	Write at pointer address
1	0	1	0	1	0	0	1	A9	Increment cursor address
1	0	1	0	1	1	0	0	AC	Read at cursor address
1	0	1	0	1	0	1	0	AA	Write at cursor address
1	0	1	0	1	1	0	1	AD	Read at cursor address and increment address
1	0	1	0	1	0	1	1	AB	Write at cursor address and increment address
1	0	1	1	1	0	1	1	BB	Write from cursor address to pointer address

NOTES:
1. Any combination of these three commands is valid.
2. Any combination of these three commands is valid.
3. d = Don't care.

Read/Write at Cursor
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor register.

Increment Cursor
Adds one (modulo 16K) to the cursor address register.

Read/Write at Cursor and Increment
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor register and then adds one (modulo 16K) to the cursor address register.

Write from Cursor to Pointer
Writes the data contained in the bus interface latch into the block of display memory designated by the cursor address and pointer address registers, inclusive. After completion of the command, the pointer address will be unchanged, but the cursor register contents will be equal to the pointer address.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		0.2			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except $\overline{INTR}$ output)	$I_{OH} = -200\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus three-state leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{OD}	$\overline{INTR}$ open drain output leakage current	$V_O = 0$ to V_{CC}			10	μA
I_{CC}	Power supply current				160	mA

NOTES:
1. Stresses above those listed under Absolute Maximum Rating may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any other condition above those in the operation section of this specification is not implied.
2. For operating at temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying voltages greater than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND).
6. Typical values are at +25°C, typical processing parameters.
7. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
8. Test condition for outputs: $C_L = 150pF$.
9. Timing is illustrated and specified to $\overline{WR}$ and $\overline{RD}$ inputs. Device may also be operated with $\overline{CE}$ as the 'strobing' input. In this case, all timing specifications apply referenced to falling and rising edges of $\overline{CE}$.
10. This specification requires that the $\overline{CE}$ input be negated (high) between read and/or write cycles.
11. $\overline{BCE}$, $\overline{WDB}$,and $\overline{RDB}$ delays track each other within 10nsec. Also, these output delays will tend to follow direction (min/max) of DADD0 – 13 delays.
12. These values were measured with a capacitance load of 150pF. To adjust the output delay,use the following correction factor: $50pF \leq C_L < 150pF$: −0.15ns/pF

AC ELECTRICAL CHARACTERISTICS T_A = 0°C to 70°C, V_{CC} = 5.0V±5%[4,5,6,7,8]

PARAMETER		TEST CONDITIONS	LIMITS				UNIT
			2.7MHz		4.0MHz		
			Min	Max	Min	Max	
Bus timing (figure 15)[9]							
t_{AS}	A0-A2 set-up time to $\overline{WR}$,$\overline{RD}$ low		30		30		ns
t_{AH}	A0-A2 hold time from $\overline{WR}$,$\overline{RD}$ high		0		0		ns
t_{CS}	$\overline{CE}$ set-up time to $\overline{WR}$,$\overline{RD}$ low		0		0		ns
t_{CH}	$\overline{CE}$ hold time from $\overline{WR}$,$\overline{RD}$ high		0		0		ns
t_{RW}	$\overline{WR}$,$\overline{RD}$ pulse width		250		250		ns
t_{DD}	Data valid after $\overline{RD}$ low			200		200	ns
t_{DF}	Data bus floating afyer $\overline{RD}$ high			100		100	ns
t_{DS}	Data set-up time to $\overline{WR}$ high		150		150		ns
t_{DH}	Data hold time from $\overline{WR}$ high		10		5		ns
t_{CC}	High time from $\overline{CE}$ to $\overline{CE}$[10]						
	Consecutive commands		600		600		ns
	Other accesses		300		300		ns
$\overline{CCLK}$ timing (figures 16 and 17)							
t_{CCP}	$\overline{CCLK}$ period		370		250		ns
t_{CCH}	$\overline{CCLK}$ high time		125		100		ns
t_{CCL}	$\overline{CCLK}$ low time		125		100		ns
	Output delay from $\overline{CCLK}$ edge[12]						ns
t_{CCD1}	DADD0-13,MBC		40	175	40	150	ns
t_{CCD2}	BLANK,HSYNC,VSYNC/CSYNC, CURSOR, $\overline{BEXT}$,$\overline{BREQ}$,$\overline{BACK}$, $\overline{BCE}$,$\overline{WDB}$, $\overline{RDB}$[11]		40	225	40	200	ns
Other timings (figures 17 and 18)							
t_{RDL}	READY/RDFLG low from $\overline{WR}$ high[9]			t_{CCP} + 30		t_{CCP} + 30	ns
t_{BAK}	$\overline{BACK}$ high from $\overline{PBREQ}$ low			225		200	ns
t_{BEXT}	$\overline{BEXT}$ high from $\overline{PBREQ}$ high			225		200	ns
t_{LPS}	Light pen strobe set-up time to $\overline{CCLK}$ low		120		120		ns
t_{LPH}	Light pen strobe hold time from $\overline{CCLK}$ low		−10		−10		ns
t_{IRL}	$\overline{INTR}$ low from $\overline{CCLK}$ low			225		200	ns
t_{IRH}	$\overline{INTR}$ high from $\overline{WR}$,$\overline{RD}$ high[9]			600		600	ns

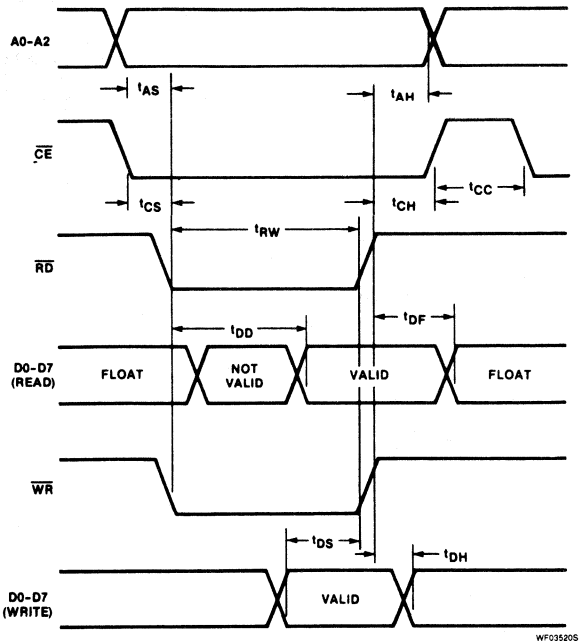
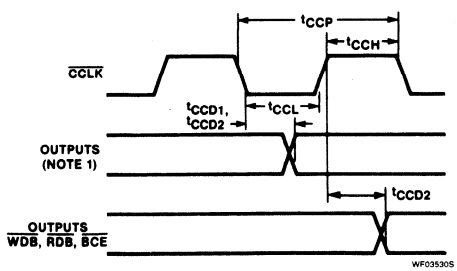


Figure 15. Bus Timing



NOTES:
1. DADD0-DADD13, BLANK, HSYNC, CSYNC/VSYNC, CURSOR, BEXT, BREQ, BCE, MBC, BACK.
2. BCE changes state on both CCLK edges —(see figures 3 and 4).

Figure 16. CCLK Timing

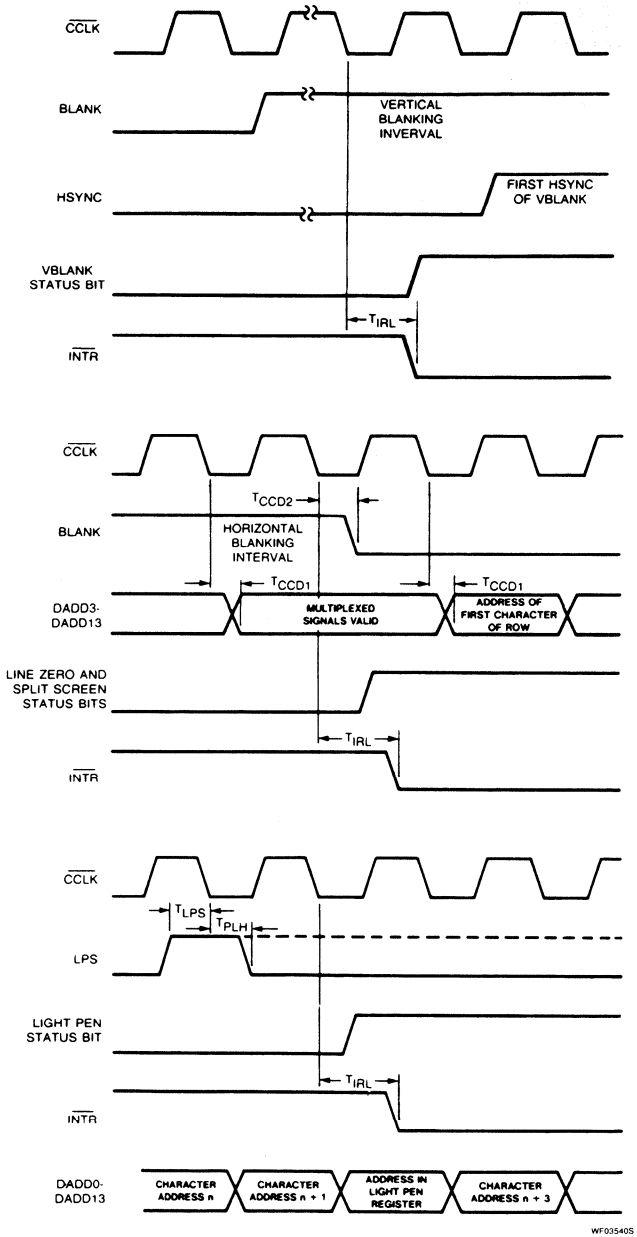


Figure 17. Other Timings

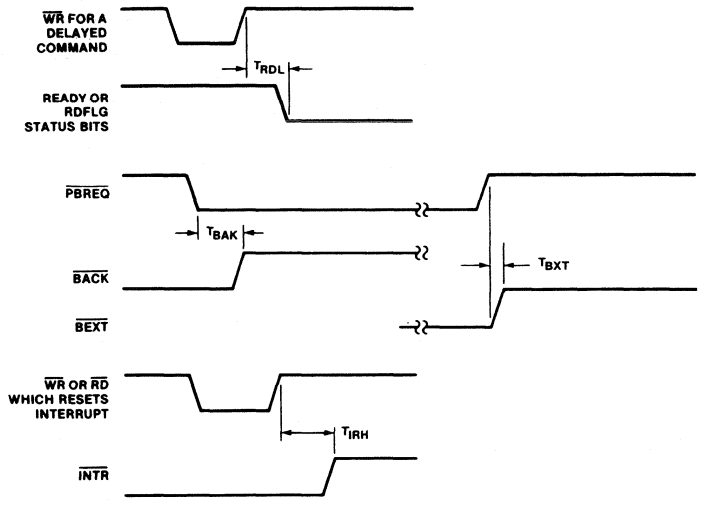
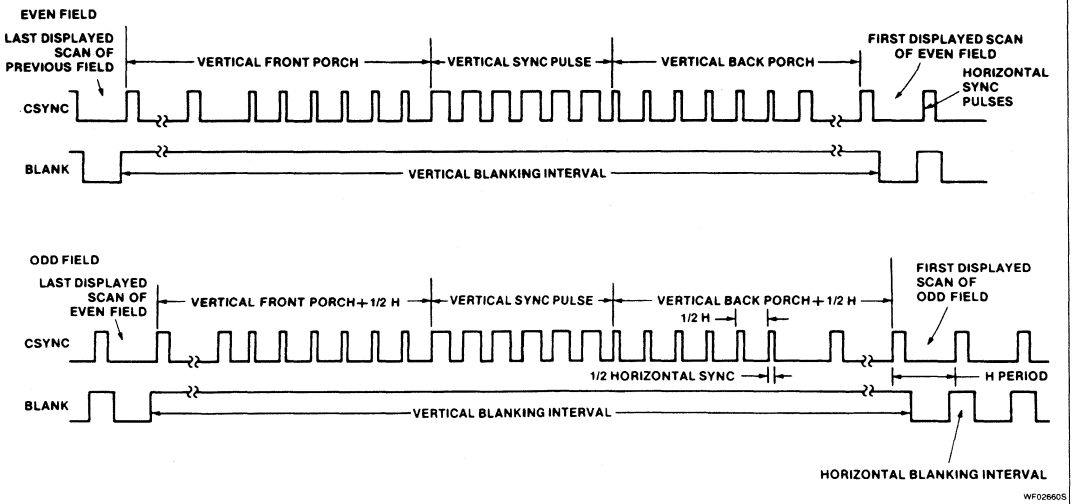


Figure 18. Other Timings



NOTES:
1. In non-interlaced operation the even field is repeated continuously, and the odd field is not.
2. In interlaced operation the even field alternates with the odd field.

Figure 19. Composite Sync Timing

Advanced Video Display Controller (AVDC)

Product Specification

Originally published by Signetics February 1985

DESCRIPTION

The Signetics SCN2674 Advanced Video Display Controller (AVDC) is a programmable device designed for use in CRT terminals and display systems that employ raster scan techniques. The AVDC generates the vertical and horizontal timing signals necessary for the display of interlaced or non-interlaced data on a CRT monitor. It provides consecutive addressing to a user specified display buffer memory domain and controls the CPU display buffer interface for various buffer configuration modes. A variety of operating modes, display formats, and timing profiles can be implemented by programming the control registers in the AVDC.

A minimum CRT terminal system configuration consists of an AVDC, an SCN2671 Keyboard and Communication Controller (PKCC), an SCN2670 Display Character and Graphics Generator (DCGG), an SCB2675 Color/Monochrome Attributes Controller (CMAC), a single chip microcomputer such as the 8048, a display buffer RAM, and a small amount of TTL for miscellaneous address decoding, interface, and control. Typically, the package count for a minimum system is between 15 and 20 devices; system complexity can be enhanced by upgrading the microprocessor and expanding via the system address and data busses.

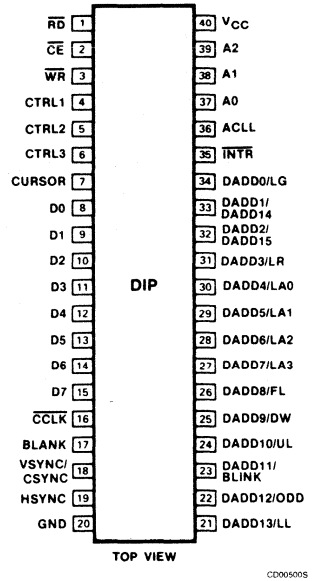
FEATURES

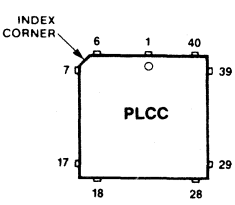
- **2.7MHz and 4MHz character rate**
- **1 to 256 characters per row**
- **1 to 16 raster lines per character row**
- **1 to 128 character rows per frame**
- **Bit mapped graphics mode**
- **Programmable horizontal and vertical sync generators**
  - **RS170 compatible sync**
- **Interlaced or non-interlaced operation**
- **Up to 64K RAM addressing for multiple page operation**
- **Readable, writable and incrementable cursor**
  - **Programmable cursor size and blink**
- **AC line lock**
- **Automatic wraparound of RAM**
- **Automatic split screen**
- **Automatic bidirectional soft scrolling**
  - **Programmable scan line increment**
- **Row table addressing mode**
- **Double height tops and bottoms**
- **Double width control output**
- **Selectable buffer interface modes**
- **Dynamic RAM refresh**
- **Completely TTL compatible**
- **Single +5 volt power supply**
- **Power on reset circuit**

APPLICATIONS

- **CRT terminals**
- **Word processing systems**
- **Small business computers**
- **Home computers**

PIN CONFIGURATION



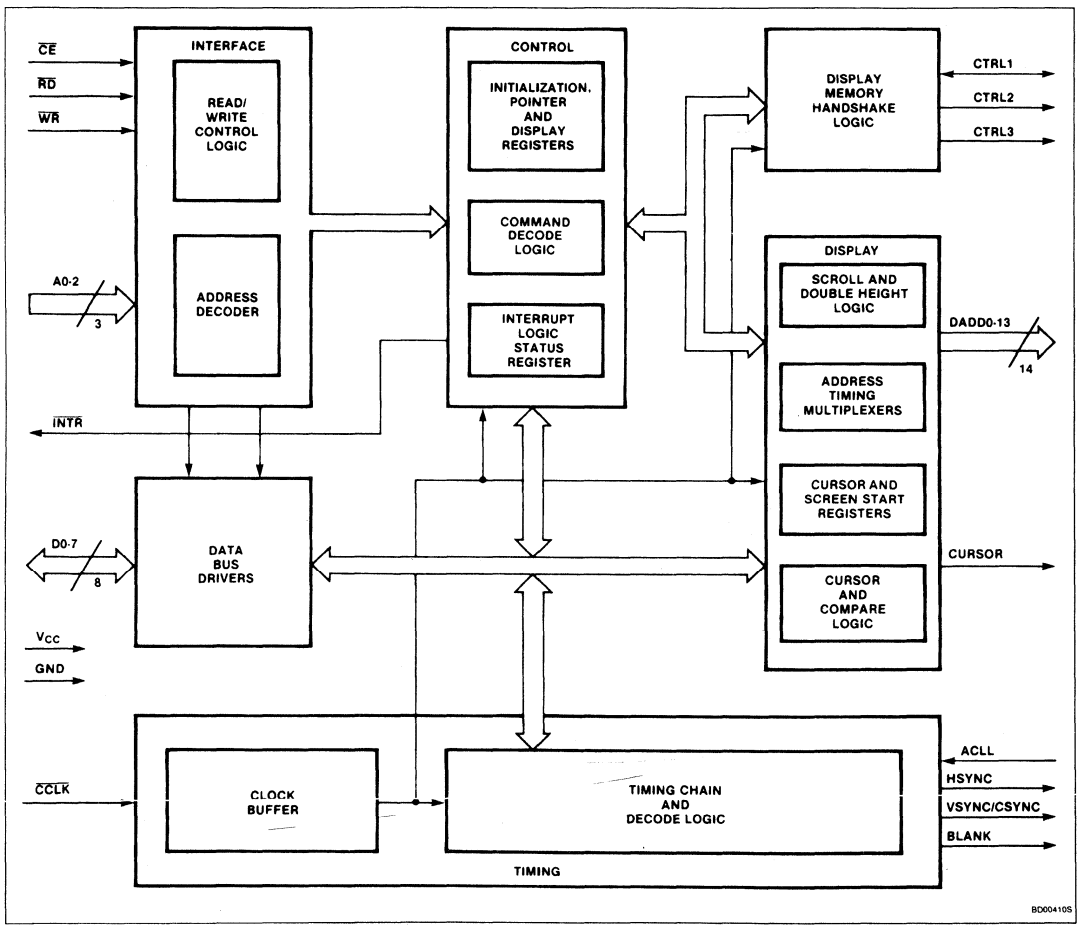


Pin	Function	Pin	Function	Pin	Function
1	NC	17	D7	31	DADD6/LA2
2	$\overline{RD}$	18	$\overline{CCLK}$	32	DADD5/LA1
3	$\overline{CE}$	19	BLANK	33	DADD4/LA0
4	$\overline{WR}$	20	VSYNC	34	NC
5	CTRL1		CSYNC/	35	DADD3/LR
6	CTRL2	21	HSYNC	36	DADD2/
7	CTRL3	22	GND		DADD15
8	CURSOR	23	NC	37	DADD1/
9	D0	24	DADD13/LL		DADD14
10	D1	25	DADD12/	38	DADD0/LG
11	D2		ODD	39	$\overline{INTR}$
12	NC	26	DADD11/	40	ACLL
13	D3		BLINK	41	A0
14	D4	27	DADD10/UL	42	A1
15	D5	28	DADD9/DW	43	A2
16	D6	29	DADD8/FL	44	V_{CC}
		30	DADD7/LA3		

ORDERING CODE

PACKAGES	V_{CC} = 5V ± 5%, T_A = 0°C to 70°C	
	4 MHz	2.7 MHz
Ceramic DIP	SCN2674BC4I40	SCN2674BC3I40
Plastic DIP	SCN2674BC4N40	SCN2674BC3N40
Plastic LCC	SCN2674BC4A44	SCN2674BC3A44

BLOCK DIAGRAM



PIN DESCRIPTION

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
A0 – A2	37 – 39	41 – 43	I	**Address Lines:** Used to select AVDC internal registers for read/write operations and for commands.
D0 – D7	8 – 15	9 – 11 13 – 17	I/O	**8-Bit Bidirectional Three-State Data Bus:** Bit 0 is the LSB and bit 7 is the MSB. All data, command, and status transfers between the CPU and the AVDC take place over this bus. The direction of the transfer is controlled by the $\overline{RD}$ and $\overline{WR}$ inputs when the $\overline{CE}$ input is low. When the $\overline{CE}$ input is high, the data bus is in the three-state condition.
$\overline{RD}$	1	2	I	**Read Strobe:** Active low input. A low on this pin while $\overline{CE}$ is low causes the contents of the register selected by A0 – A2 to be placed on the data bus. The read cycle begins on the leading (falling) edge of $\overline{RD}$.
$\overline{WR}$	3	4	I	**Write Strobe:** Active low input. A low on this pin while $\overline{CE}$ is also low causes the contents of the data bus to be transferred to the register selected by A0 – A2. The transfer occurs on the trailing (rising) edge of $\overline{WR}$.
$\overline{CE}$	2	3	I	**Chip Enable:** Active low input. When low, data transfers between the CPU and the AVDC are enabled on D0 – D7 as controlled by the $\overline{WR}$, $\overline{RD}$, and A0 – A2 inputs. When $\overline{CE}$ is high, effectively, the AVDC is isolated from the data bus and D0 – D7 are placed in the three-state condition.
$\overline{CCLK}$	16	18	I	**Character Clock:** Timing signal derived from the video dot clock which is used to synchronize the AVDC's timing functions.
HSYNC	19	21	O	**Horizontal Sync:** Active high output which provides video horizontal sync pulses. The timing parameters are programmable.
VSYNC/ CSYNC	18	20	O	**Vertical Sync/Composite Sync:** A control selects either vertical or composite sync pulses on this active high output. When CSYNC is selected, equalization pulses are included. The timing parameters are programmable.
BLANK	17	19	O	**Blank:** This active high output defines the horizontal and vertical borders of the display. Display control signals which are output on DADD0 through DADD13 are valid on the trailing edge of BLANK.
CURSOR	7	8	O	**Cursor Gate:** This output becomes active for a specified number of scan lines when the address contained in the cursor register matches the address output on DADD0 through DADD13 for displayable character addresses. The first and last lines of the cursor and a blink option are programmable. When the row table addressing mode is enabled, this output is active for a portion of the blanking interval prior to the first scan line of a character row, while the AVDC is fetching the starting address for that row.
$\overline{INTR}$	35	39	O	**Interrupt Request:** Open drain output which supplies an active low interrupt request from any of five maskable sources. This pin is inactive after a power on reset or a master reset command.
ACLL	36	40	I	**AC Line Lock:** If this input is low after the programmed vertical front porch interval, the vertical front porch will be lengthened by increments of horizontal scan line times until this input goes high. This input should be pulled high if not being used.
CTRL1	4	5	I/O	**Handshake Control 1:** In independent mode, provides an active low write data buffer ($\overline{WDB}$) output which strobes data from the interface latch into the display memory. In transparent and shared modes, this is an active low processor bus request ($\overline{PBREQ}$) input which indicates that the CPU desires to access the display memory.
CTRL2	5	6	O	**Handshake Control 2:** In independent mode, provides an active low read data buffer ($\overline{RDB}$) output which strobes data from the display memory into the interface latch. In transparent and shared modes, this is an active low bus external enable ($\overline{BEXT}$) output which indicates that the AVDC has relinquished control of the display memory (DADD0 – DADD13 are in the three-state condition) in response to a CPU bus request. $\overline{BEXT}$ also goes low in reponse to a 'display off and float DADD' command. In row buffer mode, it is an active low bus request ($\overline{BREQ}$) output which halts the CPU during a line DMA.

PIN DESCRIPTION (Continued)

MNEMONIC	PIN NO.		TYPE	NAME AND FUNCTION
	DIP	PLCC		
CTRL3	6	7	O	**Handshake Control 3:** In independent mode, provides the active low buffer chip enable ($\overline{BCE}$) signal to the display memory. In transparent and shared modes, provides an active low bus acknowledge ($\overline{BACK}$) output which serves as a ready signal to the CPU in response to a processor bus request. In row buffer mode, this is an active high memory bus control (MBC) output which configures the system for the DMA transfer of one row of character codes from system memory to the row display buffer.
DADD0–DADD13	34-21	38 – 35 33 – 24	O	**Display Address:** Used by the AVDC to address up to 16K of display memory directly, or up to 64K of memory by de-multiplexing DADD14 and DADD15. These outputs are floated at various times depending on the buffer mode. Various control signals are multiplexed on DADD0 through DADD13 and are valid at the trailing edge of BLANK. These control signals are: DADD0/LG **Line Graphics:** Output which denotes bit mapped graphic mode. DADD1/DADD14 **Display Address 14:** Multiplexed address bit used to extend addressing to 64K. DADD2/DADD15 **Display Address 15:** Multiplexed address bit used to extend addressing to 64K. DADD3/LR **Last Row:** Output which indicates the last active character row of each field. DADD4-DADD7/LA0-LA3 **Line Address:** Provides the number of the current scan line count for each character row. DADD8/FL **First Line:** Asserted during the blanking interval just prior to the first scan line of each character row. DADD9/DW **Double Width:** Output which denotes a double width character row. DADD10/UL **Underline:** Asserted during the blanking interval just prior to the scan line which matches the programmed underline position (line 0 through 15). DADD11/BLINK **Blink Frequency:** Provides an output divided down from the vertical sync rate. DADD12/ODD **Odd Field:** Active high signal which is asserted before each scan line of the odd field when interlace is specified. Replaces DADD4/LA0 as the least significant line address for interlaced sync and video applications. DADD13/LL **Last Line:** Asserted during the blanking interval just prior to the last scan line of each character row.
V_{CC}	40	44	I	**Power Supply:** +5 volts power input.
GND	20	22	I	**Ground:** Signal and power ground input.

FUNCTIONAL DESCRIPTION

As shown in the block diagram, the AVDC contains the following major blocks:

- Data bus buffer
- Interface Logic
- Operation Control
- Timing
- Display Control
- Buffer Control

Data Bus Buffer

The data bus buffer provides the interface between the external and internal data busses. It is controlled by the operation control block to allow read and write operations to take place between the controlling CPU and the AVDC.

Interface Logic

The interface logic contains address decoding and read and write circuits to permit communications with the microprocessor via the data bus buffer. The functions performed by the CPU read and write operations are shown in table 1.

Operation Control

The operation control section decodes configuration and operation commands from the CPU and generates appropriate signals to other internal sections to control the overall device operation. It contains the timing and display registers which configure the display format and operating mode, the interrupt logic, and the status register which provides operational feedback to the CPU.

Timing

The timing section contains the counters and decoding logic necessary to generate the monitor timing outputs and to control the display format. These timing parameters are

selected by programming of the initialization registers.

Display Control

The display control section generates linear addressing for up to 16K bytes of display memory. Internal comparators limit the portion of the memory which is displayed to programmed values. Additional functions performed in this section include cursor positioning and address comparisons required for generation of timing signals, double height tops and bottoms, smooth scrolling, and the split screen interrupts.

Buffer Control

The buffer control section generates three signals which control the transfer of data between the CPU and the display buffer memory. Four system configurations requiring four different 'handshaking' schemes are supported. These are described below.

SYSTEM CONFIGURATIONS

Figure 1 illustrates the block diagram of a typical display terminal using the Signetics SCN2670, SCN2671, SCN2674, and SCB2675 CRT terminal devices. In this system, the CPU examines inputs from the data communications line and the keyboard and places the data to be displayed in the display buffer memory. This buffer is typically a RAM which holds the data for a single or multiple screenload (page) or for a single character row.

The AVDC supports four common system configurations of display buffer memory: the independent, transparent, shared, and row buffer modes. The first three modes utilize a single or multiple page RAM and differ primarily in the means used to transfer display data between the RAM and the CPU. The row buffer mode makes use of a single row buffer (which can be a shift register or a small RAM) that is updated in real time to contain the appropriate display data.

The user programs bits 0 and 1 of IR0 to select the mode best suited for the system environment. The CNTRL1 – 3 outputs perform different functions for each mode and are named accordingly in the description of each mode.

Independent Mode

The CPU to RAM interface configuration for this mode is illustrated in figure 2. Transfer of data between the CPU and display memory is accomplished via a bidirectional latched port and is controlled by read data buffer ($\overline{RDB}$), write data buffer ($\overline{WDB}$), and buffer chip enable ($\overline{BCE}$). This mode provides a noncontention type of operation that does not require address multiplexers. The CPU does not address the memory directly – the read or write operation is performed at the address contained in the cursor address register or the pointer address register as specified by the CPU. The AVDC enacts the data transfers during blanking intervals in order to prevent visual disturbances of the displayed data.

Table 1. AVDC ADDRESSING

A2	A1	A0	READ ($\overline{RD}$ = 0)	WRITE ($\overline{WR}$ = 0)
0	0	0	Interrupt register	Initialization registers[1]
0	0	1	Status register	Command register
0	1	0	Screen start 1 lower register	Screen start 1 lower register
0	1	1	Screen start 1 upper register	Screen start 1 upper register
1	0	0	Cursor address lower register	Cursor address lower register
1	0	1	Cursor address upper register	Cursor address upper register
1	1	0	Screen start 2 lower register	Screen start 2 lower register
1	1	1	Screen start 2 upper register	Screen start 2 upper register

[1]There are 15 initialization registers which are accessed sequentially via a single address. The AVDC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR14) is accessed. The pointer then continues to point to IR14 for additional accesses. Upon a power-on or a master reset command, the internal pointer is reset to point to the firsr register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command.

The CPU manages the data transfers by supplying commands to the AVDC. The commands used are:

1. Read/write at pointer address.
2. Read/write at cursor address (with optional increment of address).
3. Read/write from cursor address to pointer address.

The operational sequence for a write operation is:

1. CPU checks RDFLG status bit to assure that any delayed commands have been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes address into cursor or pointer registers.
4. CPU issues 'write at cursor with/without increment' or 'write at pointer' command.
5. AVDC generates control signals and outputs specified address to perform requested operation. Data is copied from the interface latch into the memory.
6. AVDC sets RDFLG status to indicate that the write is completed.

Similarly, a read operation proceeds as follows:

1. Steps 1 and 3 as above.
2. CPU issues 'read at cursor with/without increment' or 'read at pointer' command.
3. AVDC generates control signals and outputs specified address to perform requested operation. Data is copied from memory to the interface latch and AVDC sets RDFLG status to indicate that the read is completed.
4. CPU checks RDFLG status to see if operation is completed.
5. CPU reads data from interface latch.

Loading the same data into a block of display memory is accomplished via the 'write from cursor to pointer' command:

1. CPU checks RDFLG status bit to assure that any delayed commands have been completed.
2. CPU loads data to be written to display memory into the interface latch.
3. CPU writes beginning address of memory block into cursor address register and ending address of block into pointer address register.
4. CPU issues 'write from cursor to pointer' command.
5. AVDC generates control signals and outputs block addresses to copy data from the interface latch into the specif i block of memory.
6. AVDC sets RDFLG status to indicate that the block write is completed.

Similar sequences can be implemented on an interrupt driven basis using the READY interrupt output to advise the CPU that a previously asserted delayed command has been completed.

Two timing sequences are possible for the 'read/write at cursor/pointer' commands. If the command is given during the active display window (defined as first scan line of the first character row to the last scan line of the last character row), the operation takes place during the next horizontal blanking interval, as illustrated in figure 3. If the command is given during the vertical blanking interval, or while the display has been commanded blanked, the operation takes place immediately. In the latter case, the execution time for the command is approximately five character clocks (see figure 4).

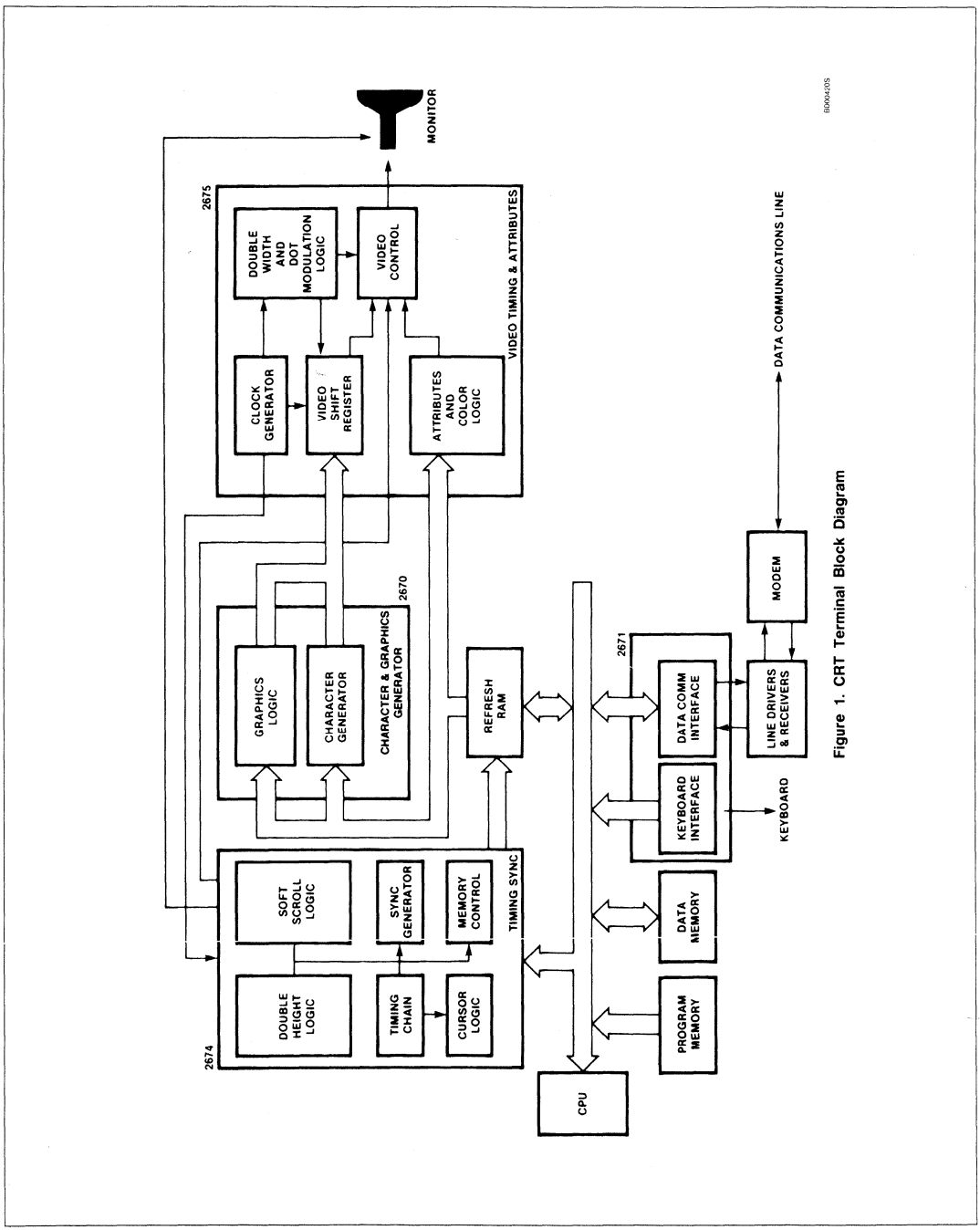


Figure 1. CRT Terminal Block Diagram

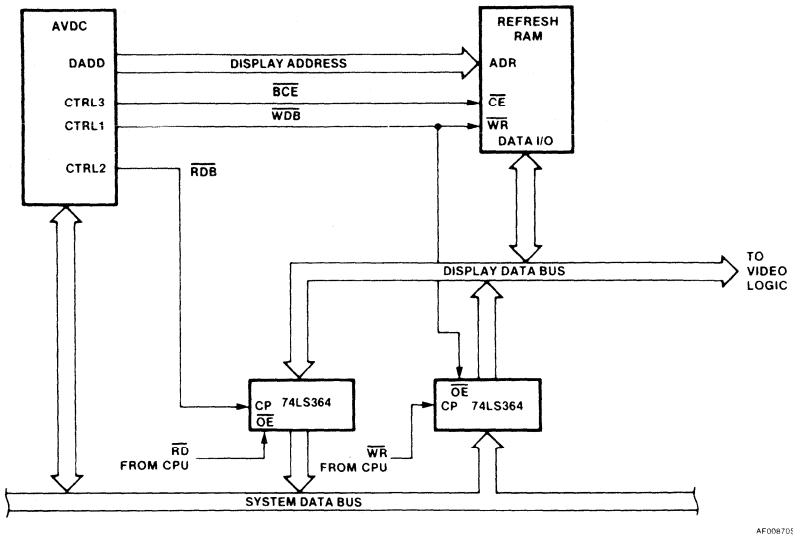
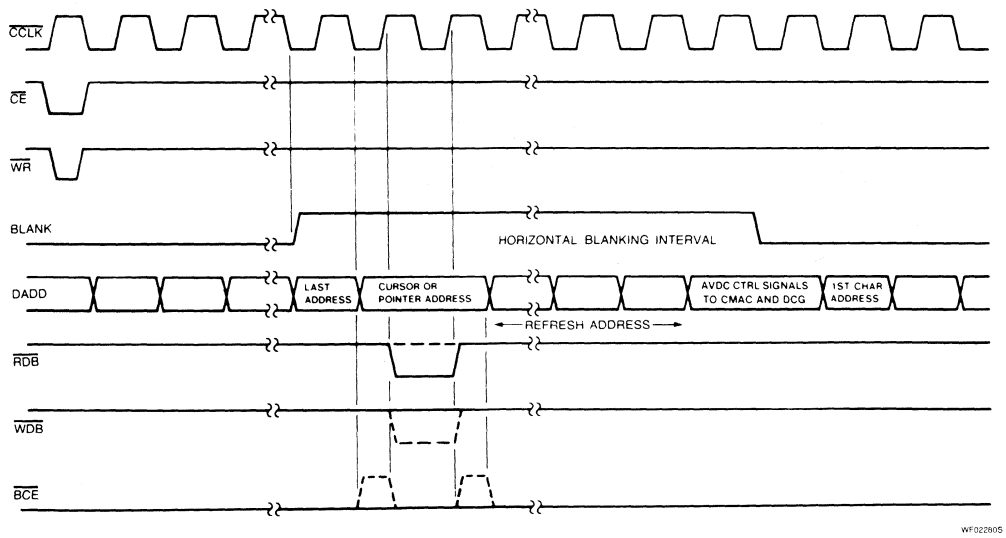


Figure 2. Independent Buffer Mode Configuration



NOTES:
1. Write waveforms shown in dotted lines.
2. If command execution occurs just prior to the first scan line of a character row and row table addressing mode is enabled, execution of the command is delayed by two character clocks from the timing illustrated

Figure 3. Read/Write at Cursor/Pointer Command Timing (Command Received During Active Display Window)

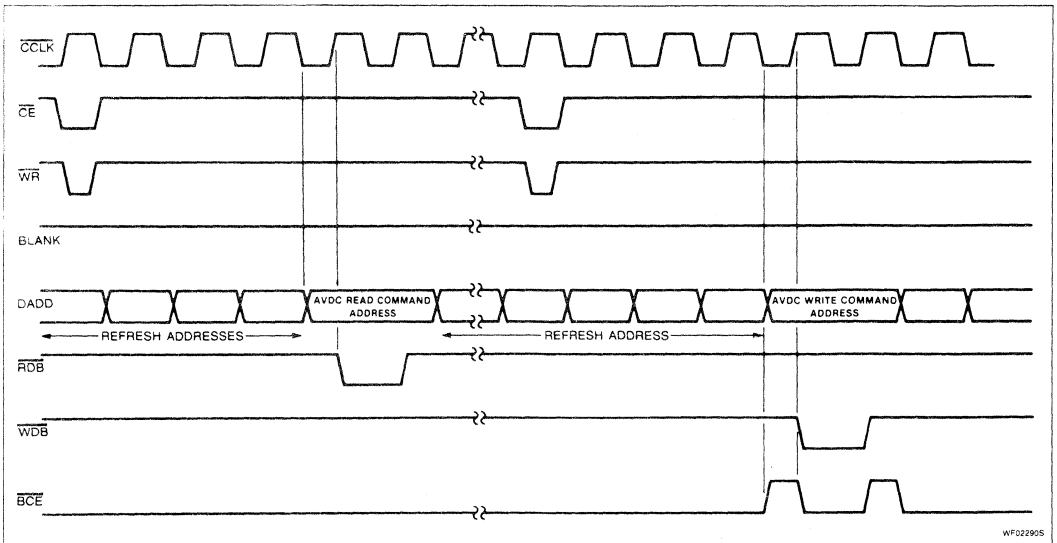


Figure 4. Read/Write at Cursor/Pointer Command Timing (Command Received While Display is Blanked)

Timing for the 'read/write from cursor to pointer' operation is shown in figure 5. The memory is filled at a rate of one location per two character times. The command will execute only during blanking intervals and may require many horizontal or vertical blanking intervals to complete. Additional delayed commands can be asserted immediately after this command has completed.

Immediate commands can be asserted at any time regardless of the state of the ready status/interrupt.

Shared and Transparent Buffer Modes

In these modes, the display buffer RAM is a part of the CPU memory domain and is addressed directly by the CPU. Both modes use the same hardware configuration with the CPU accessing the display buffer via three-state drivers (see figure 6). The processor bus request ($\overline{PBREQ}$) control signal informs the AVDC that the CPU is requesting access to the display buffer. In response to this request, the AVDC raises bus acknowledge ($\overline{BACK}$) until its bus external ($\overline{BEXT}$) output has freed the display address and data busses for CPU access. $\overline{BACK}$, which can be used as a 'hold' input to the CPU, is then lowered to indicate that the CPU can access the buffer.

In transparent mode, the AVDC delays the granting of the buffer to the CPU until a vertical or horizontal blanking interval, thereby causing minimum disturbance of the display. In shared mode, the AVDC will blank the display and grant immediate access to the CPU. Timing for these modes is illustrated in figures 7, 8, and 9.

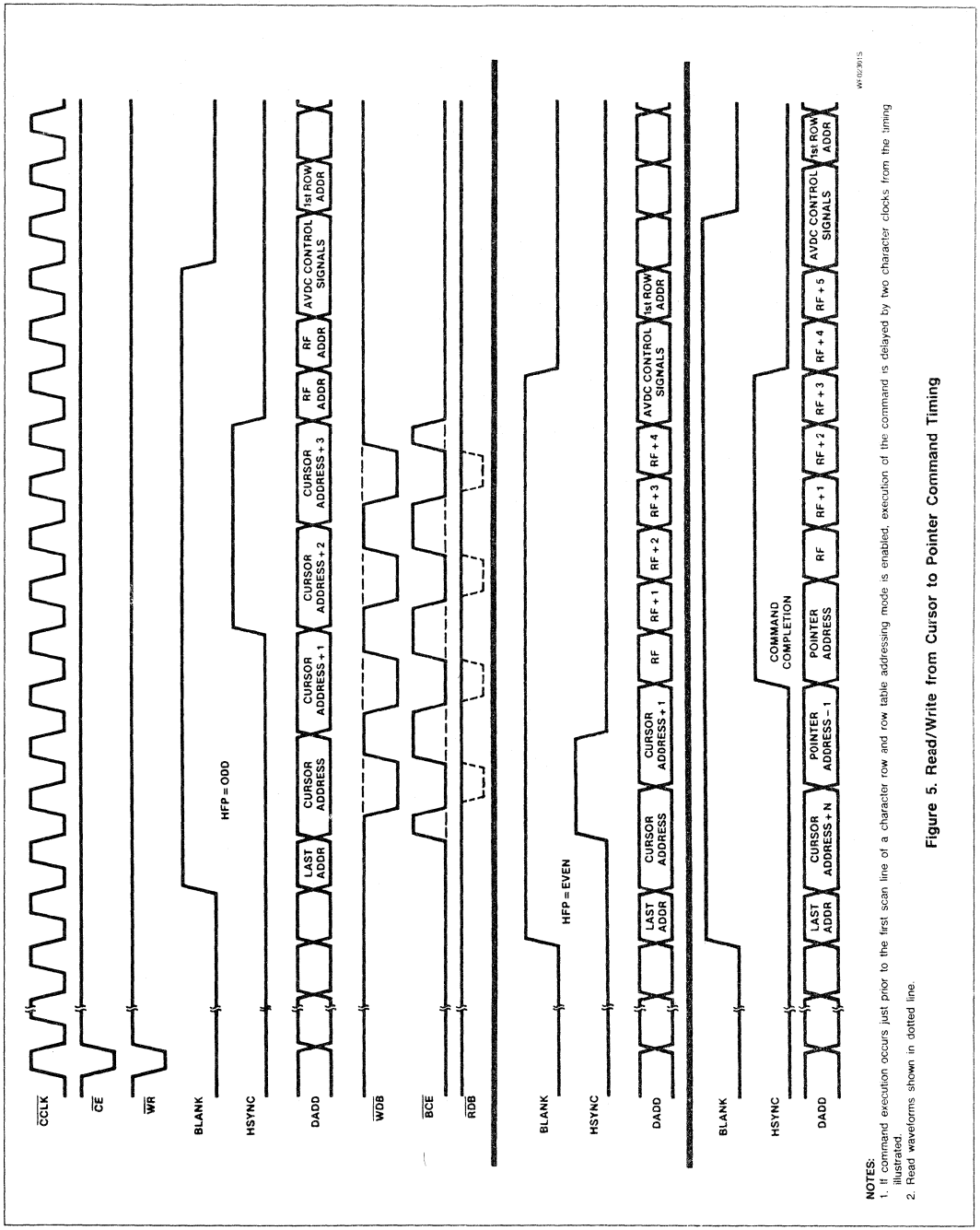
Row Buffer Mode

Figures 10 and 11 show the timing and a typical hardware implementation for the row buffer mode. During the first scan line (line 0) of each character row, the AVDC halts the CPU and DMA's the next row of character data from the system memory to the row buffer memory. The AVDC then releases the CPU and displays the row buffer data for the programmed number of scan lines. The control signal $\overline{BREQ}$ informs the CPU that character addresses and the MBC signal will start at the next falling edge of BLANK. The CPU must release the address and data busses before this time to prevent bus contention. After the row of character data is transferred to the row buffer RAM, $\overline{BREQ}$ returns high to grant memory control back to the CPU.

Row Table Addressing Mode

In this mode, each character row in the screen image memory has a unique starting address. This provides greater flexibility with respect to screen operations, such as editing, than the sequential addressing mode. The row table, figure 12, is a list of starting addresses for each character row and may reside anywhere in the AVDC's addressable memory space. Each entry in the table consists of two bytes: the first byte contains the 8 LSBs of the row starting address and the second byte contains, in its 6 least significant bits, the 6 MSBs of the row starting address. The function of the two MSBs of the second byte is selected by programming IR0[7]. They may be used either as row attribute bits to control double width and double height for that character row, or as an additional two address bits to extend the usable display memory to 64K.

The first address of the row table is designated in screen start register two (SSR2). If row table addressing is enabled via IR2[7], and the display is on, the AVDC fetches the next row's starting address from the table during the blanking interval prior to the first scan line of each character row, while simultaneously incrementing the contents of SSR2 by two so as to point to the next table entry. The fetching of the row starting address from the row table is indicated by the assertion of the CURSOR output during BLANK. The address read from the table by the AVDC is loaded into screen start register 1 (SSR1) for use internally. Since the contents of SSR2 changes as the table entries are fetched, it must be re-initialized to point to the first table entry during each vertical retrace interval.



NOTES:
1. If command execution occurs just prior to the first scan line of a character row and row table addressing mode is enabled, execution of the command is delayed by two character clocks from the timing illustrated.
2. Read waveforms shown in dotted line.

Figure 5. Read/Write from Cursor to Pointer Command Timing

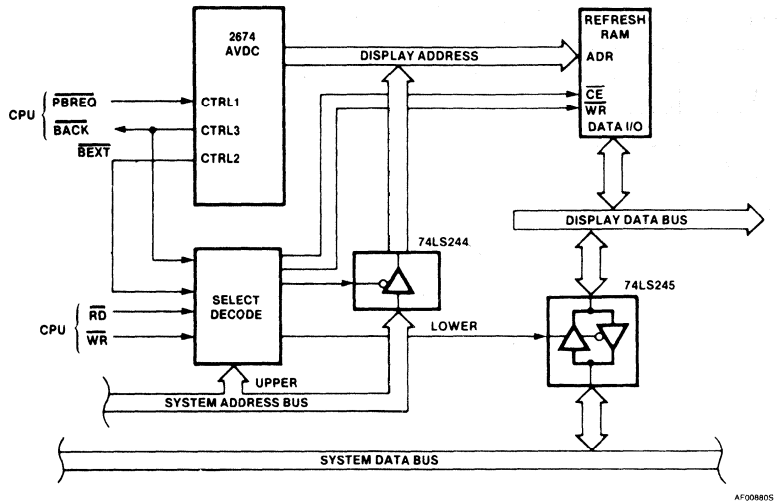
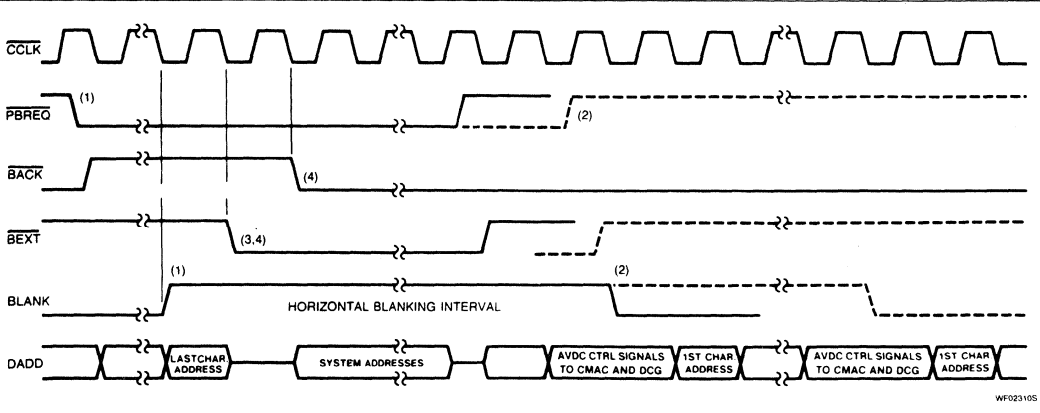
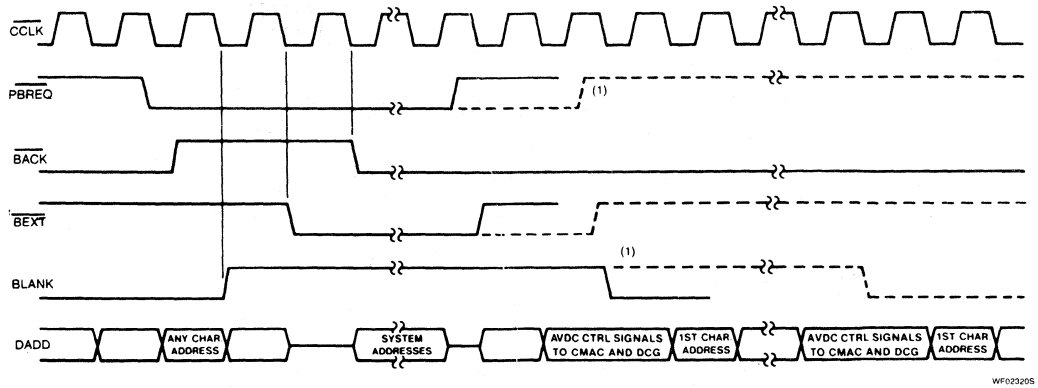


Figure 6. AVDC Shared or Transparent Buffer Modes



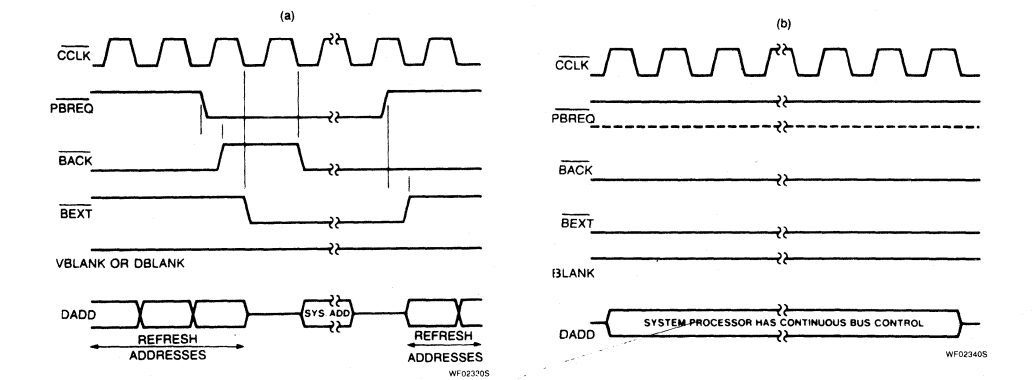
NOTES:
1. PBREQ must be asserted prior to the rising edge of BLANK in order for sequence to begin during that blanking period.
2. If PBREQ is negated after the next to last CCLK of the horizontal blanking interval, the next scan line will also be blanked.
3. Accesses during vertical blank or "display off" are granted only at the beginning of the horizontal front porch.
4. If row table addressing is enabled, CPU access is delayed by two character clocks prior to the first scan line of each character row.

Figure 7. Transparent Buffer Mode Timing



NOTE:
1. If PBREQ is negated after the next to last CCLK of the horizontal blanking interval, the next scan line will also be blanked.

Figure 8. Shared Buffer Mode Timing



a) During Vertical Blank or after 'display off' command in shared mode only. See figure 7 for transparent timing.

b) After 'display off and 3-state' command.

Figure 9. Shared and Transparent Mode Timing

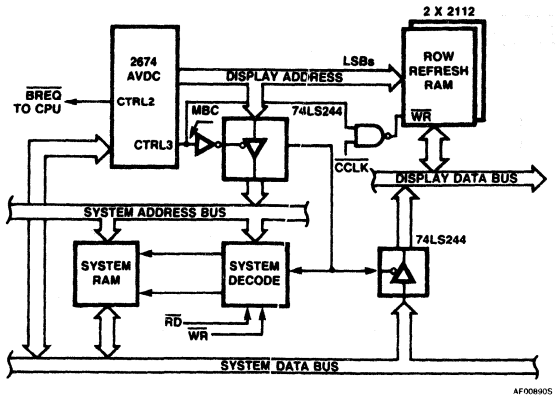
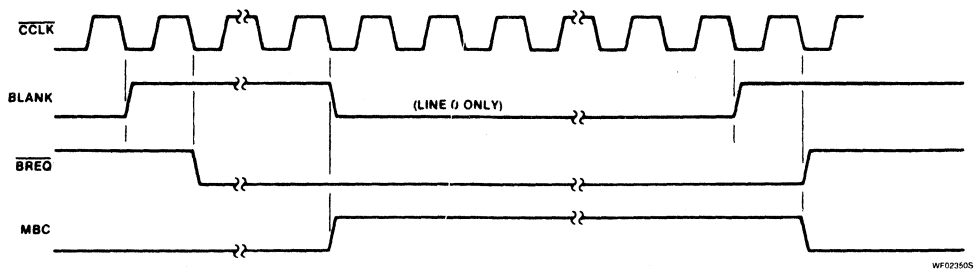


Figure 10. Row Buffer Mode Configuration



NOTE:
If row table addressing is enabled, BREQ will be asserted at the middle of the last scan line of the prior row, and MBC will be asserted at the beginning of BLANK.

Figure 11. Row Buffer Mode Timing

Row table addressing is intended primarily for use in conjunction with the row buffer mode of operation and requires no additional circuitry in that case. It may also be used with the independent and transparent buffer modes, but circuitry must be added to route the data from the display memory to the data bus inputs of the AVDC. Additionally, when not operating in row buffer mode, care must be taken to assure that the CPU does not attempt to access the AVDC while it is reading the row table. One way of preventing this is to latch the last line output which is multiplexed on DADD13 and to test this latch prior to reading or writing the AVDC. The AVDC should only be accessed if the latch is low, indicating that the last line of the row is not active.

Figure 13 illustrates a typical hardware implementation for use in conjunction with independent and transparent modes, and figure 14 shows the timing for row table operation.

OPERATION

After power is applied, the AVDC will be in an inactive state. Two consecutive 'master reset' commands are necessary to release this circuitry and ready the AVDC for operation. Two register groups exist within the AVDC: the initialization registers and the display control registers. The initialization registers select the system configuration, monitor timing, cursor shape, display memory domain, pointer address, scrolling region, double height and width condition, and screen format. These are loaded first and normally require no modification except for certain special visual effects. The display control registers specify the memory address of the base character (upper left corner of screen), the cursor position, and the split screen addresses associated with the scrolling area or an alternate memory. These may require modification during operation.

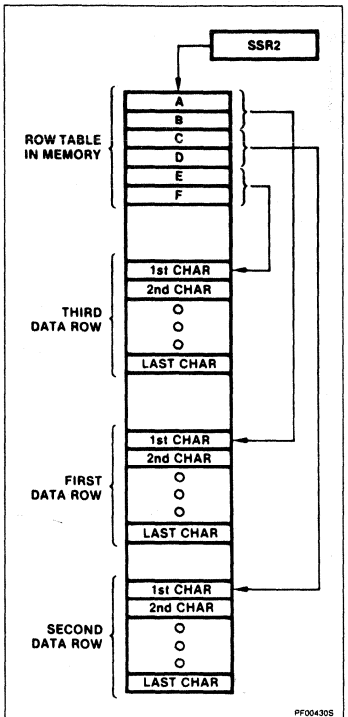


Figure 12. Row Table Address Format

After initial loading of the two register groups, the AVDC is ready to control the monitor screen. Prior to executing the AVDC commands which turn on the display and cursor, the user should load the display memory with the first data to be displayed. During operation, the AVDC will sequentially address the display memory within the limits programmed into its registers. The memory outputs character codes to the system character and graphics generation logic, where they are converted to the serial video stream necessary to display the data on the CRT. The user effects changes to the display by modifying the contents of the display memory, the AVDC display control and command registers, and the initialization registers, if required. Interrupts and status conditions generated by the AVDC supply the 'handshaking' information necessary for the CPU to effect real time display changes in the proper time frame if required.

INITIALIZATION REGISTERS

There are 15 initialization registers (IR0 – IR14) which are accessed sequentially via a single address. The AVDC maintains an internal pointer to these registers which is incremented after each write at this address until the last register (IR14) is accessed. The pointer then continues to point to IR14 for further accesses. Upon a power-on or a master reset command, the internal pointer is reset to point to the first register (IR0) of the initialization register group. The internal pointer can also be preset to any register of the group via the 'load IR address pointer' command. These registers are write only and are used to specify parameters such as the system configuration, display format, cursor shape, and monitor timing. Register formats are shown in table 2.

IR0[7] – Double Height/Width Enable
When this bit is set, the value in IR14[7:6] is used to control the double height and width conditions of each character row. Assertion of this bit also allows IR14[7:6] to be programmed in two ways:

1. By the CPU writing to IR14 directly.
2. When the contents of screen start register 1 (SSR1) upper are changed, either by the CPU writing to this register or by the automatic loading of SSR1 when operating in row table mode, the two MSBs of SSR1 upper are copied into IR14[7:6]. Thus, the MSBs of each row table entry can be used to control double height and double width attributes on a row by row basis.

IR14[5:4] are not active when this bit is set. When this bit is reset, the double height and width attributes operate as described in IR[14].

IR0[6:3] – Scan Lines per Character Row
Both interlaced and non-interlaced scanning are supported by the AVDC. For interlaced mode, two different formats can be implemented, depending on the interconnection between the AVDC and the character generator (see IR1[7]). This field defines the number of scan lines used to compose a character row for each technique. As scanning occurs, the scan line count is output on the LA0-LA3 and ODD pins.

IR0[2] – VSYNC/CSYNC Enable
This bit selects either vertical sync pulses or composite sync pulses on the VSYNC/CSYNC output (pin 18). The composite sync waveform conforms to EIA RS170 standards, with the vertical interval composed of six equalizing pulses, six vertical sync pulses, and six more equalizing pulses.

IR0[1:0] – Buffer Mode Select
Four buffer memory modes may be selectively enabled to accommodate the desired system configuration. See System Configurations.

IR1[7] – Interlace Enable
Specifies interlaced or noninterlaced timing operation. Two modes of interlaced operation are available, depending on whether LA0 – LA3 or ODD, LA0 – LA2 are used as the line address for the character generator. The resulting displays are shown in figure 15.

For 'interlaced sync' operation, the same information is displayed in both odd and even fields, resulting in enhanced readability. The AVDC outputs successive line numbers in ascending order on the LA0-LA3 lines, one per scan line for each field.

The 'interlaced sync and video' format doubles the character density on the screen. The AVDC outputs successive line numbers in ascending order on the ODD and LA0 – LA2 lines, one per scan line for each field. The number of scan lines per character row is always even. Assume that the first character row is row 0 (even). When scanning through the odd field, the scan line numbers being displayed are odd for both the even and odd character rows. When scanning through the even field, the scan line numbers being displayed are even for both even and odd character rows (see figure 15).

IR1[6:0] – Equalizing Constant
This field indirectly defines the horizontal front porch and is used internally to generate the equalizing pulses for the RS170 compatible CSYNC. The value for this field is the total number of character clocks ($\overline{CCLK}$s) during a horizontal line period divided by two, minus two times the number of character clocks in the horizontal sync pulse:

$$EC = \frac{H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}}{2} - 2(H_{SYNC})$$

The definition of the individual parameters is illustrated in figure 16. The minimum value of H_{FP} is three character clocks, four $\overline{CCLK}$'s for row table addressing. Note that when using the 2675 CMAC, it will delay the blank pulse three $\overline{CCLK}$s relative to the HSYNC pulse. Because of this delay, the actual HFP and HBP values will be different from the values programmed into the AVDC. The actual HFP will be decreased by 3 character clocks. The actual HBP will be increased by 3 character clocks.

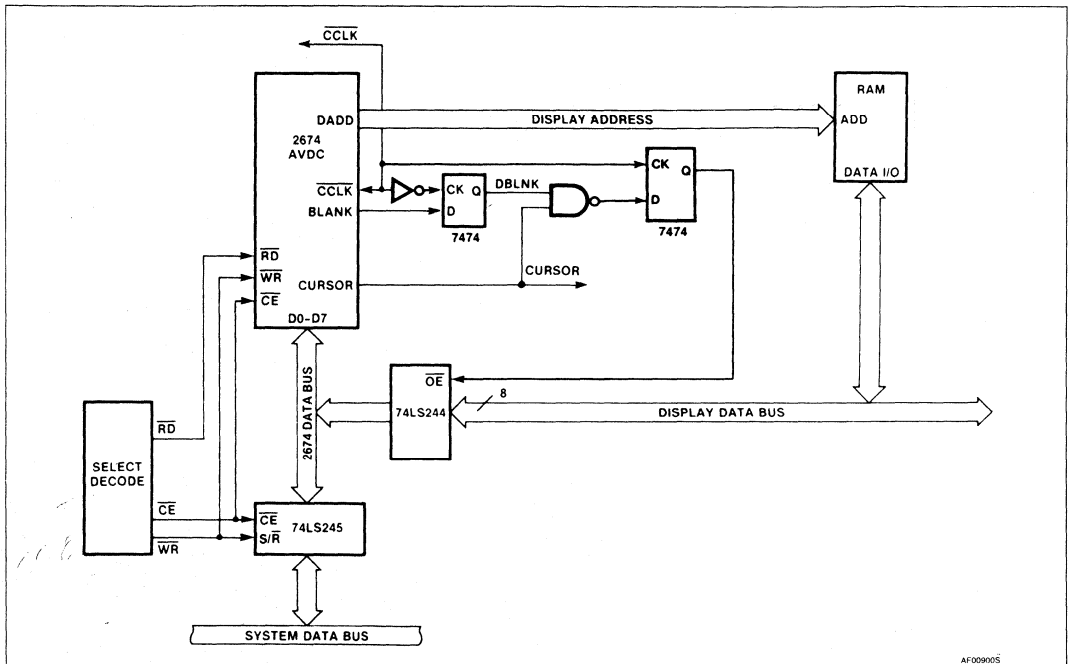


Figure 13. Row Table Mode Configuration (non-Row Buffer Modes)

IR2[7] – Row Table Mode Enable
Assertion/negation of this bit causes the AVDC to begin/terminate operating in row table mode starting at the next character row. See Row Table Addressing Mode section. By using the split interrupt capability of the AVDC, this mode can be enabled and disabled on a particular character row. This allows a combination of row table and sequential addressing to be utilized to provide maximum flexibility in generating the display.

IR2[6:3] – Horizontal Sync Pulse Width
This field specifies the width of the HSYNC pulse in $\overline{CCLK}$ periods.

IR2[2:0] – Horizontal Back Porch
This field defines the number of $\overline{CCLK}$s between the trailing edge of HSYNC and the trailing edge of BLANK.

IR3[7:5] – Vertical Front Porch
Specifies the number of scan line periods between the rising edges of BLANK and VSYNC during the vertical retrace interval. The vertical front porch will be extended in increments of scan lines if the ACLL input is low at the end of the programmed value.

IR3[4:0] – Vertical Back Porch
This field determines the number of scan line periods between the falling edges of the VSYNC and BLANK outputs.

IR4[7] – Character Blink Rate
Specifies the frequency for the character blink attribute timing. The blink rate can be specified as 1/64 or 1/128 of the vertical field rate. The timing signal has a duty cycle of 50% and is multiplexed onto the DADD11/BLINK output at the falling edge of each BLANK.

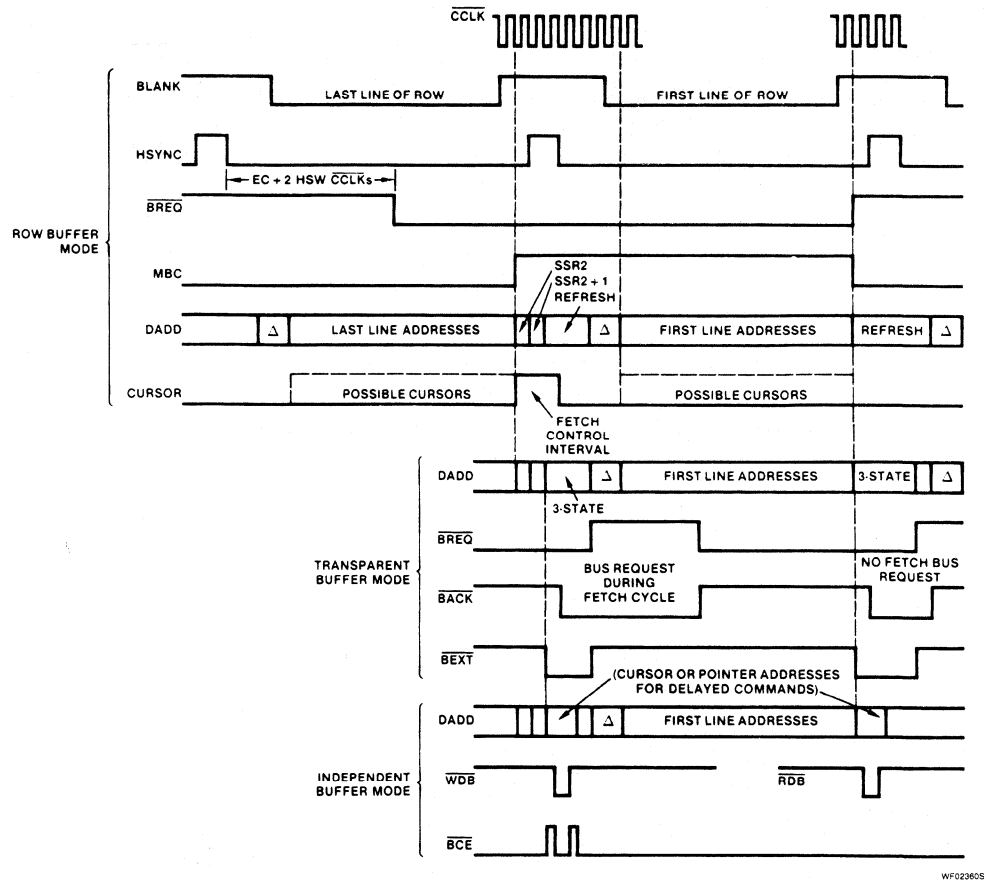
IR4[6:0] – Character Rows Per Screen
This field defines the number of character rows to be displayed. This value multiplied by the scan lines per character row, plus the vertical front porch, the vertical back porch values, and the vertical sync pulse width is the vertical scan period in scan lines.

IR5[7:0] – Active Characters Per Row
This field determines the number of characters to be displayed on each row of the CRT screen. The sum of this value, the horizontal front porch, the horizontal sync width, and the horizontal back porch is the horizontal scan period in $\overline{CCLK}$s.

IR6[7:4], IR6[3:0] – First and Last Scan Line of Cursor
These two fields specify the height and position of the cursor on the character block. The 'first' line is the topmost line when scanning from the top to the bottom of the screen. The value of the 'first line of cursor' must be less than the 'last line of cursor' value.

IR7[7:6] – Vertical Sync Pulse Width
This field specifies the width of the VSYNC pulse in scan line periods.

IR7[5] – Cursor Blink Enable
This bit controls whether or not the cursor output pin will be blinked at the selected rate (IR7[4]). The blink duty cycle for the cursor is 50%.



Δ = MULTIPLEXED CONTROL SIGNALS (LG,DW, etc.)
EC = EQUALIZING CONSTANT
HSW = HORIZONTAL SYNC WIDTH

Figure 14. Row Table Mode Timing

Table 2. INITIALIZATION REGISTER BIT FORMATS

IR0

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DOUBLE HT/WD	SCAN LINES PER CHARACTER ROW				SYNC SELECT	BUFFER MODE SELECT	
	NON-INTERLACED		INTERLACED				
0 = OFF 1 = ON	0000 = 1 LINE 0001 = 2 LINES 0010 = 3 LINES • • 1110 = 15 LINES 1111 = 16 LINES		0000 = 2 LINES 0001 = 4 LINES 0010 = 6 LINES • • 1110 = 30 LINES 1111 = UNDEFINED		0 = VSYNC 1 = CSYNC	00 = INDEPENDENT 01 = TRANSPARENT 10 = SHARED 11 = ROW	

IR1

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
INTERLACE ENABLE	EQUALIZING CONSTANT						
0 = NON-INT 1 = INTER	00000000 = 1 CCLK 00000001 = 2 CCLK • • 11111110 = 127 CCLK 11111111 = 128 CCLK		CALCULATED FROM: $EC = 0.5(H_{ACT} + H_{FP} + H_{SYNC} + H_{BP}) - 2(H_{SYNC})$				

IR2

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ROW TABLE	HORIZONTAL SYNC WIDTH				HORIZONTAL BACK PORCH		
0 = OFF 1 = ON	0000 = 2 CCLK 0001 = 4 CCLK • • 1110 = 30 CCLK 1111 = 32 CCLK				000 = NOT ALLOWED 001 = 3 CCLK • • 110 = 23 CCLK 111 = 27 CCLK		

IR3

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
VERTICAL FRONT PORCH			VERTICAL BACK PORCH				
000 = 4 SCAN LINES 001 = 8 SCAN LINES • • 110 = 28 SCAN LINES 111 = 32 SCAN LINES			00000 = 4 SCAN LINES 00001 = 6 SCAN LINES • • 11110 = 64 SCAN LINES 11111 = 66 SCAN LINES				

IR4

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
CHARACTER BLINK RATE	ACTIVE CHARACTER ROWS PER SCREEN						
0 = 1/64 VSYNC 1 = 1/128 VSYNC	00000000 = 1 ROW 00000001 = 2 ROWS • • 11111110 = 127 ROWS 11111111 = 128 ROWS						

Table 2. INITIALIZATION REGISTER BIT FORMATS (Continued)

IR5

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
ACTIVE CHARACTERS PER ROW							
00000010 = 3 CHARACTERS 00000011 = 4 CHARACTERS • • 11111110 = 255 CHARACTERS 11111111 = 256 CHARACTERS							

IR6

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
FIRST LINE OF CURSOR				LAST LINE OF CURSOR			
0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15				0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR7

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
VSYNC WIDTH		CURSOR BLINK	CURSOR RATE	UNDERLINE POSITION			
00 = 3 SCAN LN 01 = 1 SCAN LN 10 = 5 SCAN LN 11 = 7 SCAN LN		0 = OFF 1 = ON	0 = 1/32 1 = 1/64	0000 = SCAN LINE 0 0001 = SCAN LINE 1 • • 1110 = SCAN LINE 14 1111 = SCAN LINE 15			

IR8

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER FIRST ADDRESS LSB'S							
H'000 = 0 H'001 = 1 • • H'FFE = 4,094 H'FFF = 4,095				NOTE: MSB'S ARE IN IR9[3:0]			

IR9

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY BUFFER LAST ADDRESS				DISPLAY BUFFER FIRST ADDRESS MSB'S			
0000 = 1,023 0001 = 2,047 • • 1110 = 15,359 1111 = 16,383				SEE IR8			

Table 2. INITIALIZATION REGISTER BIT FORMATS (Continued)

IR10

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DISPLAY POINTER ADDRESS LOWER							
SEE IR11							

IR11

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LZ DOWN	LZ UP	DISPLAY POINTER ADDRESS UPPER					
0 = OFF 1 = ON	0 = OFF 1 = ON	H'0000' = 0 H'0001' = 1 • • H'3FFF' = 16,383					

IR12

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
SCROLL START	SPLIT REGISTER 1						
0 = OFF 1 = ON	00000000 = ROW 1 00000001 = ROW 2 • • 11111111 = ROW 128						

IR13

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
SCROLL END	SPLIT REGISTER 2						
0 = OFF 1 = ON	00000000 = ROW 1 00000001 = ROW 2 • • 11111111 = ROW 128						

IR14

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
DOUBLE 1		DOUBLE 2		LINES TO SCROLL			
00 = NORMAL 01 = DOUBLE WIDTH 10 = DB WD & TOPS 11 = DB WD & BOTS		00 = NORMAL 01 = DOUBLE WIDTH 10 = DB WD & TOPS 11 = DB WD & BOTS		0000 = 1 0001 = 2 • • 1110 = 15 1111 = 16		SCAN LINE 0 SCAN LINE 1 • • SCAN LINE 14 SCAN LINE 15	

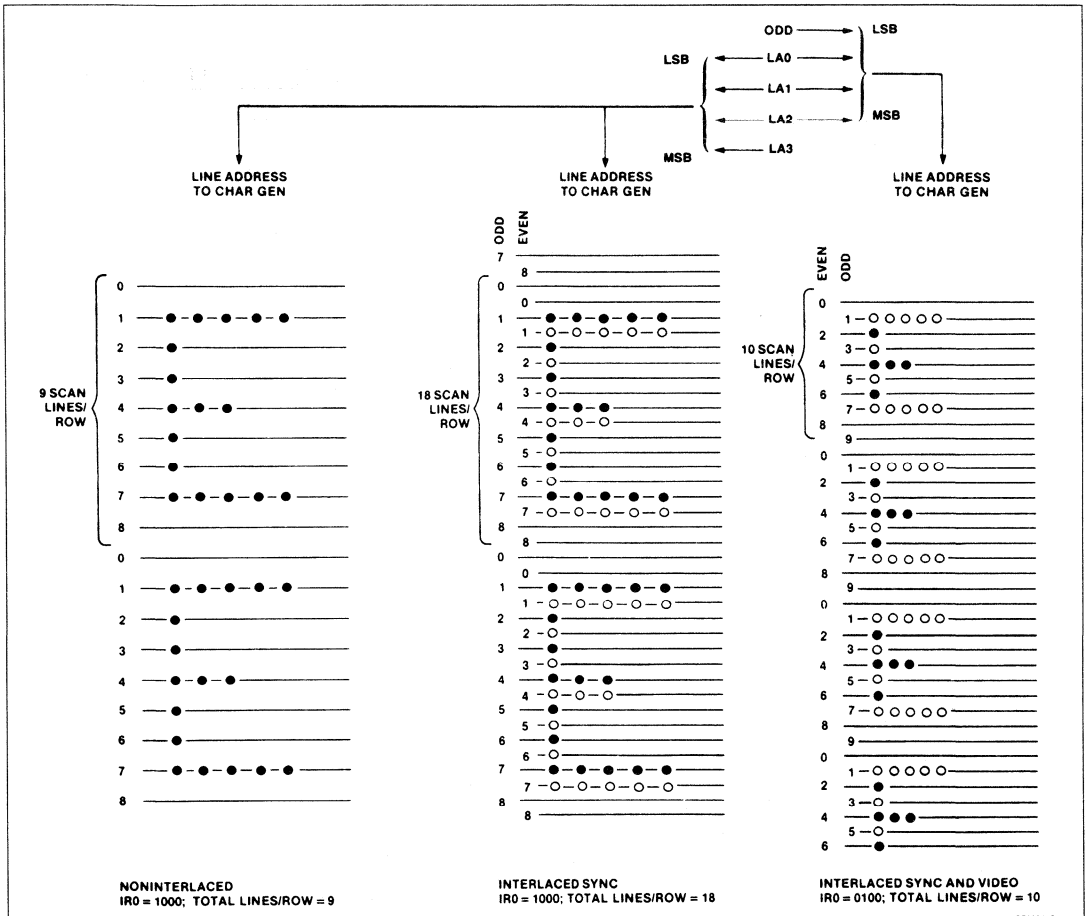


Figure 15. Interlaced Display Modes

IR7(4) – Cursor Blink Rate
The cursor blink rate can be specified at 1/32 or 1/64 of the vertical scan frequency. Blink is effective only if blink is enabled by IR7[5].

IR7[3:0] – Underline Position
This field defines which scan line of the character row will be used for the underline attribute by the 2675 CMAC. The timing signal is multiplexed onto the DADD10/UL output during the falling edge of BLANK.

IR9[3:0], IR8[7:0] – Display Buffer First Address

IR9[7:4] – Display Buffer Last Address
These two fields define the area within the buffer memory where the display data will reside. When the data at the 'display buffer last address' is displayed, the AVDC will wraparound and obtain the data to be displayed at the next screen position from the 'display buffer first address'. If 'last address' is the end of a character row and a new screen start address has been loaded into the screen start register, or if 'last address' is the last character position of the screen, the next data is obtained from the address contained in the screen start register.

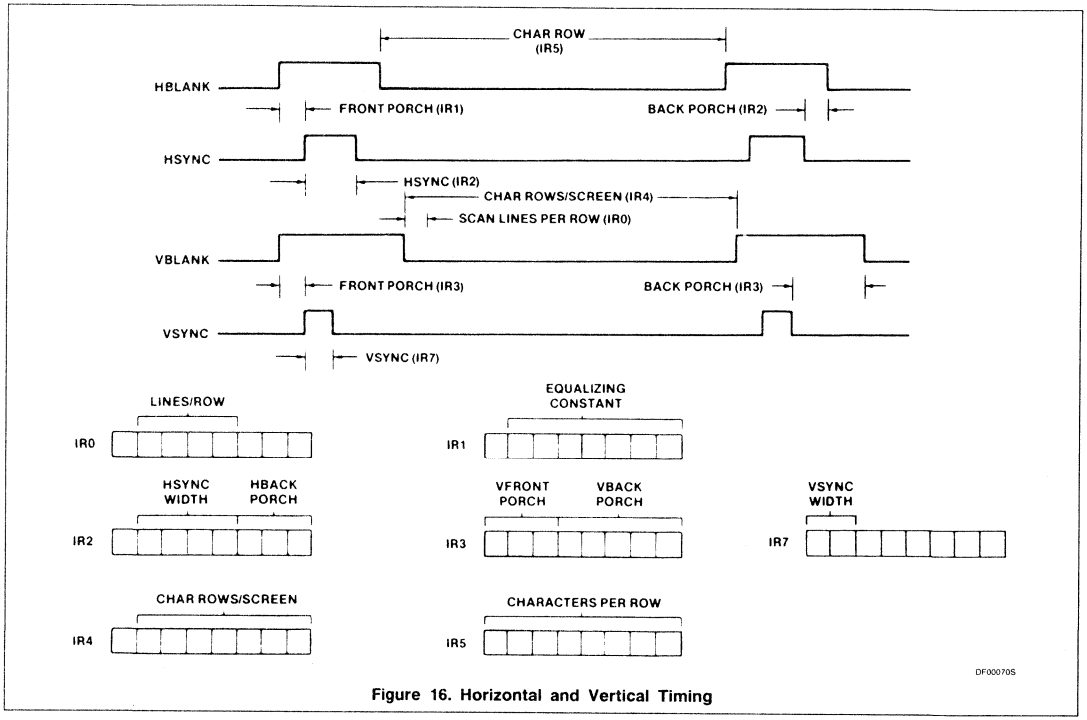


Figure 16. Horizontal and Vertical Timing

Note that there is no restriction in displaying data from other areas of the addressable memory. Normally, the area between these two bounds is used for data which can be overwritten (e.g., as a result of scrolling), while data that is not to be overwritten would be contained outside these bounds and accessed by means of the automatic split screen or split screen interrupt feature of the AVDC.

IR10[7:0] – Display Pointer Address Lower

IR11[5:0] – Display Pointer Address Upper

These two fields define a buffer memory address for AVDC controlled accesses in response to 'read/write at pointer' commands. They also define the last buffer memory address to be written for the 'write from cursor to pointer' command.

In the independent mode, the RDFLG bit of the status register should be checked for the ready state (bit 5 equal to a logic one) before writing to the display pointer address registers. Checking the status register will prevent the pointer address from being changed while a delayed command (e.g. write from cursor to pointer) is still being executed.

IR11[7] – Scan Line Zero During Scroll Down

During a scroll down operation, the new character row will appear at the top of the scrolling region. If this bit is set (logic one), the scan line count pins (LA0 through LA3) will be forced to zero for every scan line of the partial row. If the character generator provides blanks for scan line zero, the new row being scrolled into the screen will be blanked. This feature can be used to blank the new row to give the CPU time to load the new data in the display buffer. When this bit is set to a logic zero, the new data will be displayed.

IR11[6] – Scan Line Zero During Scroll Up

During a scroll up operation, the new character row will appear at the bottom of the scrolling region. If this bit is set (logic one), the scan line count pins (LA0 through LA3) will be forced to zero for every scan line of the partial row. If the character generator provides blanks for scan line zero, the new row being scrolled into the screen will be blanked. This feature can be used to blank the new row to give the CPU time to load the new data in the display buffer. When this bit is set to a logic zero, the new data will be displayed.

IR12[7] – Scroll Start

This bit is asserted when soft scroll is to take place. The scrolling area begins at the row specified in split register 1 (IR12[6:0]). If set, the first row to scroll scan line count will be reduced by the value in the lines to scroll register (IR14[3:0]). The scan line count of this row will start at the programmed offset value. When this bit is asserted, scroll end IR13[7] must be set before split 2.

IR12[6:0] – Split Register 1

Split register 1 can be used to provide special screen effects such as soft (scan line by scan line) scrolling, double height/width rows, or to change the normal addressing sequence of the display memory. The contents of this field are compared, in real time, to the current row number. Upon a match, the AVDC sets the split screen 1 status bit, and issues an interrupt request if so programmed. The status change/interrupt request is made at the beginning of scan line zero of the split screen character row. If enabled by the SPL1 bit of screen start register 2, an automatic split screen to the address specified in screen start register 2 will be made for the designated character row. During a scroll operation,

this field defines the first character row of the scrolling area.

IR13[7] – Scroll End
This field specifies that the row programmed in split register 2 (IR13[6:0]) is to be the last scrolling row of the scrolling area. Note that this bit must be asserted for a valid row only when the scroll start bit IR12[7] is also asserted.

IR13[6:0] – Split Register 2
This field is similar to the split register 1 field except for the following:

1. Split screen 2 status bit is set.
2. During a scroll operation, this field defines the last character row of the scrolling area. This row will be followed by a partial row. The LTSR (IR14) value replaces the normal scan lines/row value for the partial row, thus keeping the total scan lines/screen the same.
3. If enabled by the SPL2 bit of screen start register 2, an automatic split to the address contained in screen start register 2 will occur in one of two ways: a) If not scrolling an automatic split will occur for the next character row. b) If scrolling, the automatic split will occur after the partial row being scrolled onto or off the screen.
4. The specified double width and height conditions (IR14) are also asserted in two possible ways: a) Automatic split will assert the programmed condition for the current row. b) During soft scroll operation the programmed conditions are asserted for the partial row scrolling onto or off the screen.

IR14[7:6] – Double 1
This field specifies the conditions (double width/height or normal) of the row designated in split register 1 (IR12[6:0]). When double height tops or bottoms has been specified, the AVDC will automatically toggle between tops and bottoms until another split 1 or 2 occurs which changes the double height/width condition. If a double height tops row is specified, the scan line count will start at zero and increment the scan line count every other scan line. If a double height bottom row is specified, the AVDC will start at one half the normal scan line total. If double width is specified, the AVDC will assert the DADD9/DW output at the falling edge of blank. This condition will also remain active until the next split 1 or 2. When IR0[7] = 1, the values written into bits 7 and 6 of screen start 1 upper will also be written into IR14[7:6] and the automatic toggling between tops and bottoms is disabled.

The AVDC still addresses the RAMs on a single width character basis. The clock rate of the AVDC does not change. The display RAMs must have data as if two single wide characters are to be displayed. The first data bits addressed by the AVDC will specify the double wide character. The next data bits addressed are not displayed. The 2675 CMAC will ignore the second clock cycle data when the ADOUBLE pin is high.

IR14[5:4] – Double 2
This field specifies the conditions (double width/height or normal) of the row designated in split register 2 (IR13[6:0]). Not used when IR0[7] = 1.

IR14[3:0] – Lines to Scroll
This field defines the scan line increment to be used during a soft scroll operation. These 4 bits control the scroll rate. When smooth scrolling up by scan line increments of one, the initial value is 0000 (scan line 0) and is increased every vertical frame according to the number of lines per character row. When smooth scrolling down by scan line increments of one, the initial value is 1110 hex (scan line 14, assuming 15 scan lines per character row) and is decreased by one every vertical frame. This value will only be used when scroll start (IR12[7]) and scroll end (IR13[7]) are enabled.

Timing Considerations
Normally, the contents of the initialization registers are not changed during normal operation. However, this may be necessary to implement special display features such as multiple cursors and horizontal scrolling. Table 3 describes timing details for these registers which should be considered when implementing these features.

Display Control Registers

There are seven registers in this group, each with an individual address. Their formats are illustrated in table 4. The command register is used to invoke one of 19 possible AVDC commands as described in the COMMANDS section of this data sheet. The remaining registers in the group store address values which specify the cursor location, the location of the first character to be displayed on the screen, and any split screen address locations. The user initializes these registers after powering on the system and changes their values to control the data which is displayed.

Screen Start Registers 1 and 2
The screen start 1 registers contain the address of the first character of the first row (upper left corner of the active display). At the beginning of the first scan line of the first row, this address is transferred to the row start register (RSR) and into the memory address counter (MAC). The counter is then advanced sequentially at the character clock rate for the number of times programmed into the active characters per row register (IR5), thus reaching the address of the last character of the row plus one. At the beginning of each subsequent scan line of the first row, the MAC is reloaded from the RSR and the above sequence is repeated. At the end of the last scan line of the first row, the contents of the MAC are loaded into the RSR to serve as the starting memory address for the second character row. This process is repeated for the programmed number of rows per screen. Thus, the data in the display memory is displayed sequentially starting from the address contained in the screen start register. After the ensuing vertical retrace interval, the contents of the screen start registers are reloaded into the RSR and MAC, and the process is repeated.

During vertical blanking, the address counter operation is modified by stopping the automatic load of the contents of the RSR into the counter, thereby allowing the address outputs to free-run. This allows dynamic memory refresh to occur during the vertical retrace interval. The refresh addressing starts at the last address displayed on the screen and increments by one for each character clock during the retrace interval. If the display buffer last address is encountered, wraparound will occur. Refreshing will continue from the display buffer first address. In the independent mode, the refresh addressing will occur if no delayed commands are being executed. In the transparent and shared modes, refresh will occur during the blanking interval unless the CPU has control of the display address bus. In the row buffer mode, refresh will occur during all blanking intervals except for the first character clock time in the BLANK after the first scan line (scan line 0) of a character row.

The sequential addressing operation described above will be modified upon the occurrence of the following:

1. After reaching the 'display buffer last address.'
2. Rewriting the contents of the screen start 1 registers.
3. Setting the split register 1 or split register 2 bits of screen start register 2 upper.
4. Enabling the row table addressing mode.

First, if during the incrementing of the memory address counter the 'display buffer last address' (IR9[7:4]) is reached, the MAC will be loaded from the 'display buffer first address' register (IR9[3:0] and IR8[7:0]) at the next character clock. Sequential operation will then resume starting from this address. This wraparound operation allows portions of the display buffer to be used for purposes other than storage of displayable data and is completely automatic without any CPU intervention (see figure 17a).

Second, if the contents of screen start register 1 (upper, lower, or both) are changed during any character row (e.g., row 'n'), the starting address of the next character row (row 'n + 1') will be the new value of the screen start register and addressing will con-

tinue sequentially from there. This allows features such as split screen operation, partial scroll, or status line display to be implemented. The split screen interrupt feature of the AVDC is useful in controlling the CPU initiated operations. Note that in order to obtain the correct screen display, screen start register 1 must be reloaded with the original (origin of display) value prior to the end of the vertical retrace. See figure 17b.

The screen start two registers contain a 14-bit display address. The SSR2 address is implemented on the occurrence of item 3 above. If bit 6 of SSR2 upper is set, the SSR2 contents will be automatically loaded into the RSR at the beginning of the first scan line of the row designated by split register 1 (IR12[6:0]). If bit 7 of SSR2 upper is set, the SSR2 contents will be automatically loaded into the RSR at the beginning of the first scan line of the row specified by split register 2 (IR13[6:0]). SPL1 and SPL2 are write only bits and will read as zero when reading screen start register 2. If these bits are not used, they should be set to zeros after power up.

Lastly, when row table addressing mode is enabled, the first address of the row table is designated in SSR2. The AVDC fetches the next row's starting address from the table during the blanking interval prior to the first scan line of each character row and loads it into SSR1 for use as the starting address of the next row. Since the contents of SSR2 changes as the table entries are fetched, it must be re-initialized to point to the first table entry during each vertical retrace interval.

The values in the two MSBs of SSR1 upper are multiplexed onto the DADD1/DADD14 and DADD2/DADD15 outputs during the falling edge of BLANK. If IR0[7] = 0, these two bits act as memory page select bits which may be used to extend the display memory addressing range of the AVDC up to 64K. In that case, these two bits act as a two-bit counter which is incremented each time that 'wraparound' occurs (see above). Note that the counter is incremented at the falling edge of BLANK and that for proper display operation the wraparound address should be programmed to occur at the last character position of a row. Also, the first address accessed in the new page will be the address contained in the display buffer first address register (IR9[3:0] and IR8[7:0]). DADD14 and DADD15 should only be used in the bit mapped graphics mode.

Table 3. TIMING CONSIDERATIONS

PARAMETER	TIMING CONSIDERATIONS
First line of cursor Last line of cursor Underline line	These parameters must be established at a minimum of two character times prior to their occurrence
Double height character rows Double width character rows Rows to scroll	Set/reset prior to the row specified in split 1 or 2 registers
Cursor blink Cursor blink rate Character blink rate	New values become effective within one field after values are changed
Split register 1 Split register 2	Change anytime prior to line zero of desired row
Character rows per screen	Change only during vertical blanking period
Vertical front porch	Change prior to first line of VFP
Vertical back porch	Change prior to fourth line after VSYNC
Screen start register 1 Row table mode enable	Change prior to the horizontal blanking interval of the last line of character row before row where new value is to be used

Cursor Address Registers

The contents of these registers define the buffer memory address of the cursor. The cursor output will be asserted when the memory address counter matches the value of the cursor address registers for the scan lines specified in IR6. The cursor address registers can be read or written by the CPU or incremented via the 'increment cursor address' command. In independent buffer mode, these registers define a buffer memory address for AVDC controlled access in response to 'read/write at cursor with/without increment' commands, or the first address to be used in executing the 'write from cursor to pointer' command.

In the independent mode, the RDFLG bit of the status register should be checked for the ready state (bit 5 equal to a logic one) before writing to the cursor address registers. Checking the status register will prevent the cursor address from being changed while a cursor delayed command (e.g., write from cursor to pointer) is still being executed.

Interrupt/Status Registers

The interrupt and status registers provide information to the CPU to allow it to interact with the AVDC to effect desired changes that implement various display operations. The interrupt register provides information on five possible interrupting conditions, as shown in Table 5. These conditions can be selectively enabled or disabled (masked) from causing interrupts by certain AVDC commands. An interrupt condition which is enabled (mask bit equal to one) will cause the $\overline{INTR}$ output to be asserted and will cause the corresponding bit in the interrupt register to be set upon the occurrence of the interrupting condition. An interrupt condition which is disabled (mask bit equal to zero) has no effect on either the $\overline{INTR}$ output or the interrupt register.

The status register provides six bits of status information: the five possible interrupting conditions plus the RDFLG bit. For this register, however, the contents are not affected by the state of the mask bits.

Table 4. DISPLAY CONTROL REGISTER BIT FORMATS

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
COMMAND CODE							
SEE COMMANDS SECTION FOR COMMAND CODES							

COMMAND REGISTERS (WRITE ONLY)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
UPPER REGISTER							
Not used			MSB's				

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LOWER REGISTER (LBS's)							
H'0000' = 0 H'0001' = 1 THRU H'3FFE' = 16,382 H'3FFF' = 16,383							Note: MSB's are in Upper Register [5:0]

NOTE:
1. Bits 7 and 6 of upper register are not used in the cursor address register.

CURSOR ADDRESS REGISTERS (READ AND WRITE)

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
UPPER REGISTER							
DADD15	DADD14		MSB's				

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LOWER REGISTER (LSB)							
H'0000' = 0 H'0001' = 1 THRU H'3FFE' = 16,382 H'3FFF' = 16,383							NOTE: MSB's ARE IN UPPER REGISTER [5:0]

SCREEN START 1 REGISTERS (READ AND WRITE)

NOTES:
2. Bits 7 and 6 of upper register are always zero when read by the CPU.
3. When IR0(7) = 1, the values written into bits 7 and 6 of screen start 1 upper will also be written into IR14(7:6) to control the double width and double height attributes of the display as follows:

7	6	Attribute
0	0	None
0	1	Double width only
1	0	Double width and double
1	1	Double width and double

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
UPPER REGISTER							
SPL2 0 = OFF 1 = ON	SPL1 0 = OFF 1 = ON		MSB's				

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
LOWER REGISTER							
H'0000' = 0 H'0001' = 1 THRU H'3FFE' = 16,382 H'3FFF' = 16,383							NOTE: MSB's ARE IN UPPER REGISTER [5:0]

SCREEN START 2 REGISTERS (READ AND WRITE)

NOTES:
1. Bit 7 and bit 6 are always zero when read by CPU.
2. These bits should be set to zero after power up by the user.

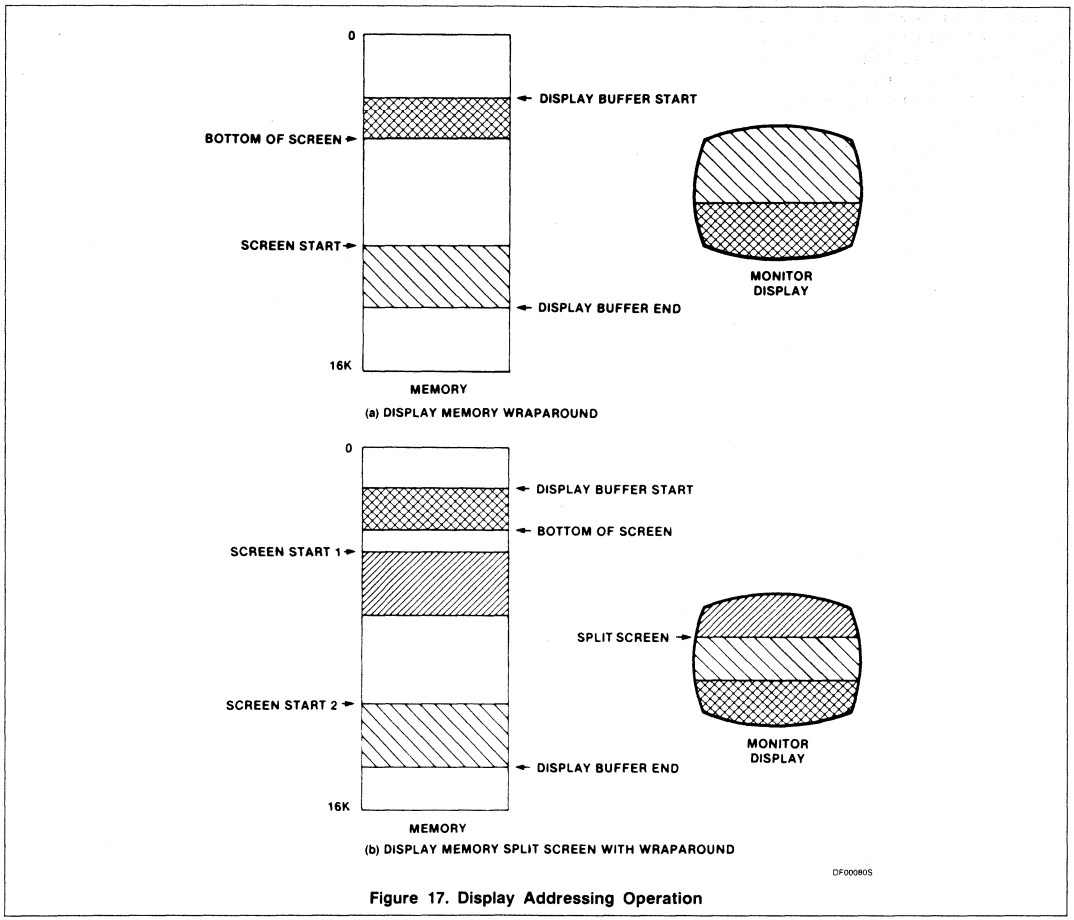


Figure 17. Display Addressing Operation

Table 5. INTERRUPT AND STATUS REGISTER BIT FORMATS

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
NOT USED ALWAYS READ AS 0		RDFLG	VBLANK	LINE ZERO	SPLIT 1	READY	SPLIT 2
		0 = BUSY 1 = READY *	0 = NO 1 = YES	0 = NO 1 = YES	0 = NO 1 = YES	0 = BUSY 1 = READY	0 = NO 1 = YES

* STATUS REGISTER ONLY. ALWAYS 0 WHEN READING INTERRUPT REGISTER.

Descriptions of each interrupt/status register bit follow. Unless otherwise indicated, a bit, once set, will remain set until reset by the CPU by issuing a 'reset interrupt/status bits' command. The bits are also reset by a 'master reset' command and upon power-up.

SR[5] – RDFLG
This bit is present in the status register only. A zero indicates that the AVDC is currently executing the previously issued delayed command. A one indicates that the AVDC is ready to accept a new delayed command. This bit is set to a one upon a master reset.

I/SR[4] – VBLANK
Indicates the beginning of a vertical blanking interval. Set to one at the beginning of the first scan line of the vertical front porch.

I/SR[3] – Line Zero
Set to one at the beginning of the first scan line (line 0) of each active character row.

I/SR[2] – Split Screen 1
This bit is set when a match occurs between the current character row number and the value contained in split register 1, IR12[6:0]. The equality condition is only checked at the beginning of line zero of each character row.

I/SR[1] – Ready
The delayed commands affect the display and may require the AVDC to wait for a blanking interval before enacting the command. This bit is set to one when execution of a delayed command has been completed. No other delayed command should be invoked until the prior delayed command is completed. This bit is set to a zero upon a master reset.

I/SR[0] – Split Screen 2
This bit is set when a match occurs between the current character row number and the value contained in split register 2 (IR13[6:0]) when you are not scrolling. It is set for the value contained in (split screen register 2) + 1 when scrolling.

COMMANDS

The AVDC commands are divided into two classes: the instantaneous commands which are executed immediately after they are invoked, and the delayed commands which may need to wait for a blanking interval prior to their execution. Command formats are shown in table 6. The commands are asserted by performing a write operation to the command register with the appropriate bit pattern as the data byte.

Instantaneous Commands

The instantaneous commands are executed immediately after the trailing edge of the $\overline{WR}$ pulse during which the command is issued. These commands do not affect the state of the RDFLG or READY interrupt/status bits and can be invoked at any time.

Master Reset
This command initializes the AVDC and can be invoked at any time to return the AVDC to its initial state. Upon power-up, two successive master reset commands must be applied to release the AVDC's internal power 'on circuits. In transparent and shared buffer modes, the CNTRL1 input must be high when the command is issued. The command causes the following:

1. VSYNC and HSYNC are driven low for the duration of the command and BLANK goes high. After command completion, HSYNC and VSYNC will begin operation and BLANK will remain high until a 'display on' command is received.
2. The interrupt and status bits and masks are set to zero, except for the RDFLG flag which is set to a one.
3. The row buffer mode, cursor-off, display-off, and the line graphics disable status are set.
4. The initialization register pointer is set to address IR0.
5. IR2[7] is reset.

Load IR Address
This command is used to preset the initialization register pointer with the value 'V' defined by D3 – D0. Allowable values are 0 to 14.

Enable Graphics
After invoking this command, the AVDC will increment the MAC to the next consecutive memory address for each scan line even if more than one scan line per row is programmed. In other words, each scan line begins with a consecutive address from the last displayed address of the previous scan line. This mode can be used for bit-mapped graphics where each location in the display buffer within the defined area contains the bit pattern to be displayed. The graphics mode will be enabled on the next character row after the 'graphics enable' command has been executed. This allows the user to enter and exit graphics mode on character row boundaries.

To perform split screen operations while in graphics mode use SSR2 only.

DADD0/LG is asserted during the trailing edge of BLANK for each scan line while this mode is active.

The AVDC allows up to 128 character rows (initialization register 4) by 256 characters (initialization register 5). For a higher resolution bit mapped screen, the AVDC is programmed as if there are characters and character rows. For example, screen size of 240 x 512 pixels is possible by programming the AVDC for 20 rows with 12 scan lines per row by 64 characters with 8 dots per character.

In the graphics mode, SSR1 should only be updated during the last scan line of the defined 'character row.'

The bit mapped graphics mode will work only in the independent and transparent modes. Row table addressing will be available as an option with the graphics mode in version T of the AVDC.

Disable Graphics
Normal addressing resumes at the next row boundary.

Display Off
Asserts the BLANK output. The DADD0 through DADD13 display address bus outputs can be optionally placed in the three-state condition by setting bit 2 to a '1' when invoking the command.

Display On
Restores normal blanking operation either at the beginning of the next field (bit 2 = 1) or at the beginning of the next scan line (bit 2 = 0). Also returns the DADD0 – DADD13 drivers to their active state.

Cursor Off
Disables cursor operation. Cursor output is placed in the low state.

Cursor On
Enables normal cursor operation.

Reset Interrupt/Status Bits
This command resets the designated bits in the interrupt and status registers. The bit positions correspond to the bit positions in the registers:

Bit 0 – Split 2

Bit 1 – Ready

Bit 2 – Split 1

Bit 3 – Line zero

Bit 4 – Vertical blank

Disable Interrupts
Sets the interrupt mask to zeros for the designated conditions, thus disabling these conditions from being set in the interrupt register and asserting the $\overline{INTR}$ output. Bit position correspondence is as above.

Enable Interrupts
This command writes the associated interrupt mask bits to a one. This enables the corresponding conditions to be set in the interrupt register and asserts the $\overline{INTR}$ output. Bit position correspondence is as above.

Delayed Commands

This group of commands is utilized for the independent buffer mode of operation, although the 'increment cursor' command can also be used in other modes. With the excep-

Table 6. AVDC COMMAND FORMATS

D7	D6	D5	D4	D3	D2	D1	D0		COMMAND
Instantaneous Commands:									
0	0	0	0	0	0	0	0		Master reset
0	0	0	1	V	V	V	V		Load IR pointer with value V (V = 0 to 14)
0	0	1	d	d	d	1	0[1]		Disable graphics
0	0	1	d	d	d	1	1[2]		Enable graphics
0	0	1	d	1	N	d	0[1]		Display off. Float DADD bus if N = 1
0	0	1	d	1	N	d	1[2]		Display on: Next field (N = 1) or scan line (N = 0)
0	0	1	1	d	d	d	0[1]		Cursor off
0	0	1	1	d	d	d	1[2]		Cursor on
0	1	0	N	N	N	N	N		Reset interrupt/status: Bit reset where N = 1
1	0	0	N	N	N	N	N		Disable interrupt: Disable where N = 1
0	1	1	N	N	N	N	N		Enable interrupt: Enables interrupts where N = 1
			V B	L Z	S P 1	R D Y	S P 2		**Interrupt Bit Assignments**
Delayed Commands:								**Hex**	
1	0	1	0	0	1	0	0	A4	Read at pointer address
1	0	1	0	0	0	1	0	A2	Write at pointer address
1	0	1	0	1	0	0	1	A9	Increment cursor address
1	0	1	0	1	1	0	0	AC	Read at cursor address
1	0	1	0	1	0	1	0	AA	Write at cursor address
1	0	1	0	1	1	0	1	AD	Read at cusor address and increment address
1	0	1	0	1	0	1	1	AB	Write at cusor address and increment address
1	0	1	1	1	0	1	1	BB	Write from cursor address to pointer address
1	0	1	1	1	1	0	1	BD	Read from cursor address to pointer address

NOTES:
1. Any combination of these three commands is valid.
2. Any combination of these three commands is valid.
3. d = don't care.

ion of the 'write from cursor to pointer' and increment cursor' commands, all the commands of this type will be executed immediately or will be delayed depending on when the command is invoked. If invoked during the active screen time, the command is executed at the next horizontal blanking interval. If invoked during a vertical retrace interval or a 'display off' state, the command is executed immediately.

The 'increment cursor' command is executed immediately after it is issued and requires approximately three $\overline{CCLK}$ periods for completion. The 'write from cursor to pointer' command executes during blanking intervals. The AVDC will execute as many writes as possible during each blanking interval. If the command is not completed during the current blanking interval, the command will be held in suspension during the next active portion of the screen and continues during the next blanking interval until the command is completed.

In all cases, the AVDC will assert the READY/RDFLG status to signify completion of the delayed command. No other delayed command should be given until the previous delayed command has completed. Therefore, the READY interrupt or RDFLG status flag should be used for handshaking control between the AVDC and CPU when using the delayed commands.

Read/Write at Pointer
Transfers data between the display buffer and the bus interface latch using the address contained in the pointer registers.

Read/Write at Cursor
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor registers.

Increment Cursor
Adds one (modulo 16K) to the cursor address registers.

Read/Write at Cursor and Increment
Transfers data between the display buffer and the bus interface latch using the address contained in the cursor registers and then adds one (modulo 16K) to the cursor address registers.

Write from Cursor to Pointer
Writes the data contained in the bus interface latch into the block of display memory designated by the cursor address and pointer address registers, inclusive. After completion of the command, the pointer address will be unchanged, but the cursor register contents will be equal to the pointer address.

Read From Cursor to Pointer
Writes the data from the block of display memory designated by the cursor and pointer addresses inclusive into the bus interface latch. This command can be used for a DMA dump of memory into RAM from the cursor location to the pointer location. After completion of the command, the cursor register contents will equal the pointer register contents.

ABSOLUTE MAXIMUM RATINGS[1]

PARAMETER	RATING	UNIT
Operating ambient temperature[2]	0 to +70	°C
Storage temperature	−65 to +150	°C
All voltages with respect to ground[3]	−0.5 to +6.0	V

DC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6]

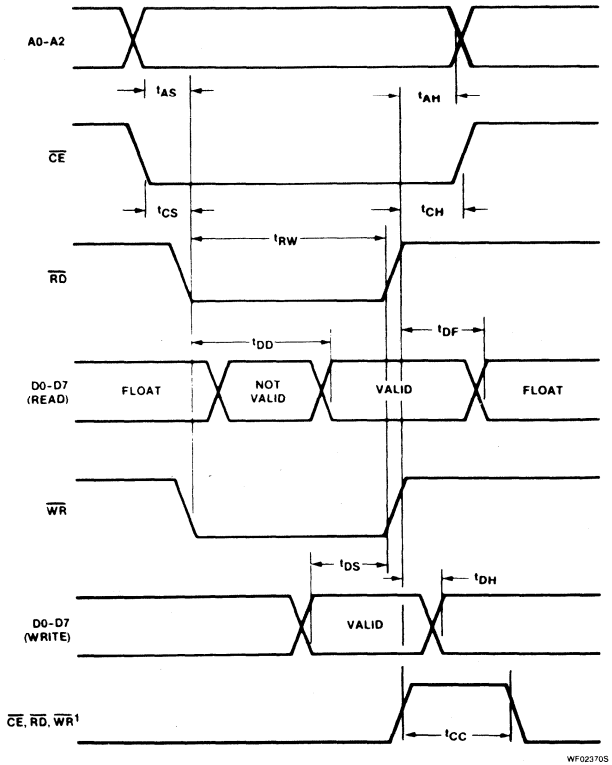
PARAMETER		TEST CONDITIONS	LIMITS			UNIT
			Min	Typ	Max	
V_{IL}	Input low voltage				0.8	V
V_{IH}	Input high voltage		2.0			V
V_{OL}	Output low voltage	$I_{OL} = 2.4mA$			0.4	V
V_{OH}	Output high voltage (except $\overline{INTR}$ output)	$I_{OH} = -200\mu A$	2.4			V
I_{IL}	Input leakage current	$V_{IN} = 0$ to V_{CC}	−10		10	μA
I_{LL}	Data bus 3-state leakage current	$V_O = 0$ to V_{CC}	−10		10	μA
I_{OD}	$\overline{INTR}$ open drain output leakage current	$V_O = 0$ to V_{CC}			10	μA
I_{CC}	Power supply current				185	mA

AC ELECTRICAL CHARACTERISTICS $T_A = 0°C$ to $+70°C$, $V_{CC} = 5.0V \pm 5\%$[4,5,6,7]

PARAMETER		TEST CONDITIONS[8]	2.7MHz		4.0MHz		UNIT
			Min	Max	Min	Max	
Bus timing (figure 18)[9]							
t_{AS}	A0 – A2 set-up time to $\overline{WR}$, $\overline{RD}$ low		30		30		ns
t_{AH}	A0 – A2 hold time from $\overline{WR}$, $\overline{RD}$ high		0		0		ns
t_{CS}	$\overline{CE}$ set-up time to $\overline{WR}$, $\overline{RD}$ low		0		0		ns
t_{CH}	$\overline{CE}$ hold time from $\overline{WR}$, $\overline{RD}$ high		0		0		ns
t_{RW}	$\overline{WR}$, $\overline{RD}$ pulse width		250		200		ns
t_{DD}	Data valid after $\overline{RD}$ low			200		200	ns
t_{DF}	Data bus floating after $\overline{RD}$ high			100		100	ns
t_{DS}	Data set-up time to $\overline{WR}$ high		150		150		ns
t_{DH}	Data hold time from $\overline{WR}$ high		10		5		ns
t_{CC}	High time from $\overline{CE}$ to $\overline{CE}$						
	Consecutive commands		t_{CCP}		t_{CCP}		ns
	Other accesses		300		300		ns
CCLK timing (figure 19, 20, 21)							
t_{CCP}	$\overline{CCLK}$ period		370	10,000	250	10,000	ns
t_{CCH}	$\overline{CCLK}$ high time		125		100		ns
t_{CCL}	$\overline{CCLK}$ low time		125		100		ns
	Output delay from $\overline{CCLK}$ edge[11]						
t_{CCD1}	DADD0 – 13, MBC		40	175	40	150	ns
t_{CCD2}	BLANK, HSYNC, VSYNC/CSYNC, CURSOR, $\overline{BEXT}$, $\overline{BREQ}$, $\overline{BACK}$, $\overline{BCE}$, $\overline{WDB}$, $\overline{RDB}$[10]		40	225	40	200	ns
Other timings (figure 20)							
t_{RDL}	READY/RDFLG low from $\overline{WR}$ high[9]			$t_{CCP}+30$		$t_{CCP}+30$	ns
t_{BAK}	$\overline{BACK}$ high from $\overline{PBREQ}$ low			225		200	ns
t_{BXT}	$\overline{BEXT}$ high from $\overline{PBREQ}$ high			225		200	ns
t_{IRL}	$\overline{INTR}$ low from $\overline{CCLK}$ low			225		200	ns
t_{IRH}	$\overline{INTR}$ high from $\overline{WR}$, $\overline{RD}$ high[9]			600		600	ns
t_{AC}	ACLL from HSYNC		$3xt_{CCP}$		$3xt_{CCP}$		ns
Row table input timing (figure 21)							
t_{DSRT}	Data set-up time to $\overline{CCLK}$ low		100		60		ns
t_{DHRT}	Data hold time from $\overline{CCLK}$ low		60		60		ns

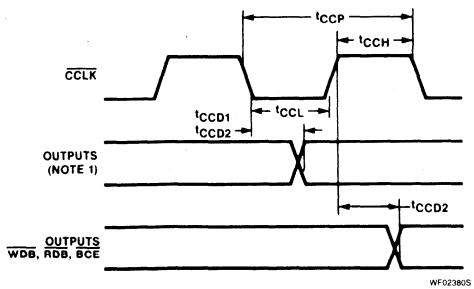
NOTES:

1. Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or at any conditions above those in the operation section of this specification is not implied.
2. For operating at elevated temperatures, the device must be derated based on +150°C maximum junction temperature.
3. This product includes circuitry specifically designed for the protection of its internal devices from damaging effects of excessive static charge. Nonetheless, it is suggested that conventional precautions be taken to avoid applying any voltages larger than the rated maxima.
4. Parameters are valid over specified temperature range.
5. All voltage measurements are referenced to ground (GND).
6. Typical values are at +25°C, typical supply voltages, and typical processing parameters.
7. For testing, all input signals swing between 0.4V and 2.4V with a transition time of 20ns maximum. All time measurements are referenced at input voltages of 0.8V and 2.0V and output voltages of 0.8V and 2.0V as appropriate.
8. Test condition for outputs: $C_L = 150pF$.
9. Timing is illustrated and specified referenced to $\overline{WR}$ and $\overline{RD}$ inputs. Device may also be operated with $\overline{CE}$ as the 'strobing' input. In this case, all timing specifications apply referenced to falling and rising edges of $\overline{CE}$.
10. $\overline{BCE}$, $\overline{WDB}$, and $\overline{RDB}$ delays track each other within 10nsec. Also, these output delays will tend to follow direction (min/max) of DADD0 – 13 delays.
11. These values were measured with a capacitance load of 150pF. To adjust the output delay, use the following correction factor: $50pF \leq C_L < 150pF$: −0.15ns/pF



NOTE:
1. Any two must be high for t_{CC}.

Figure 18. Bus Timing



NOTES:
1. DADD0 - DADD13, BLANK, HSYNC, CSYNC/VSYNC, CURSOR, $\overline{BEXT}$, $\overline{BREQ}$, $\overline{BCE}$, MBC, $\overline{BACK}$.
2. $\overline{BCE}$ CHANGES STATE ON BOTH $\overline{CCLK}$ EDGES — (see figures 3 and 4).

Figure 19. $\overline{CCLK}$ Timing

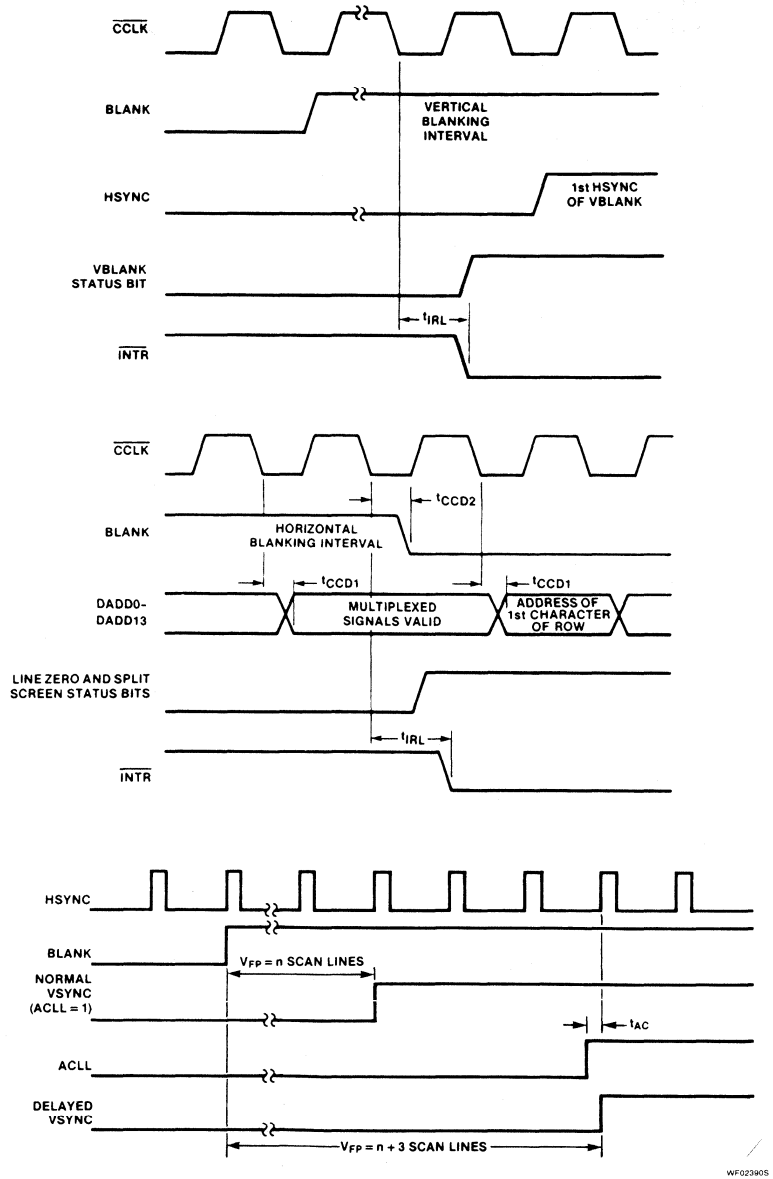


Figure 20. Other Timings

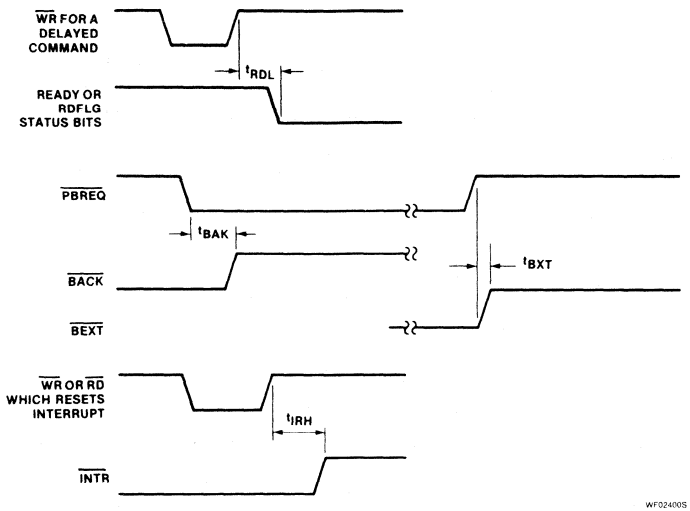


Figure 20. Other Timings (Continued)

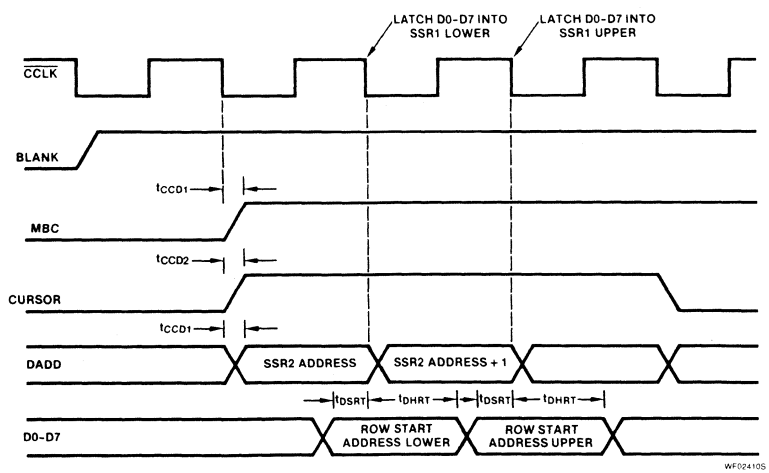
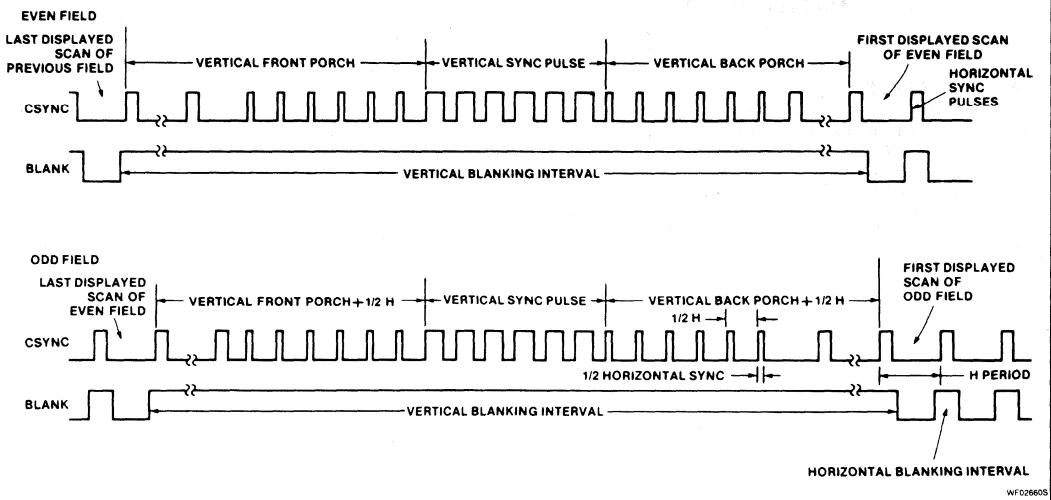


Figure 21. Row Table Fetch I/O Timing



NOTES:
1. In non-interlaced operation the even field is repeated continuously.
2. In interlaced operation the even field alternates with the odd field.

Figure 22. Composite Sync Timing

USING THE 2670/71/72/73 CRT TERMINAL CHIP SET

INTRODUCTION

Microprocessors and LSI have had a dramatic impact on the implementation and capabilities of alphanumeric CRT terminals. The first generation of CRT terminals were little more than 'glass teletypes'. Current designs, implemented with microprocessors, are characterized by an abundance of sophisticated features that were previously not economically feasible: a universal hardware design that can adapt to different user requirements simply by changing software or firmware; programmability to provide end users with the flexibility to execute specialized routines; and local intelligence and storage which off-loads the host CPU by permitting data manipulation and verification at the terminal site.

Just as the impact of microcomputers has been felt in the functional capabilities of terminals, advances in semiconductor technology have revolutionized the hardware implementation. Designs that previously consisted of 100 to 200 ICs can now be realized with a few dozen MSI and LSI devices. The majority of the LSI manufacturers' effort with respect to CRT terminals has been concentrated in the 'CRT controller' area. These circuits provide the character timing, display addressing, and sync generation functions required by all terminals. However, these controllers need to be supported by many other external circuits to implement a complete terminal.

The purpose of this application note is to provide information on the use of four new Signetics CRT terminal products which, when combined with standard CPUs, memories, and TTL, allow the implementation of a wide spectrum of CRT terminal capabilities in as few as 15 total packages. These devices are:

- 2670 Display Character and Graphics Generator (DCGG)
- 2671 Programmable Keyboard and Communications Controller (PKCC)
- 2672 Programmable Video Timing Controller (PVTC)
- 2673 Video and Attributes Controller (VAC)

MAJOR ELEMENTS OF A CRT TERMINAL

Figure 1 shows the major elements of a typical low-end microcomputer-based CRT terminal. In this system, the CPU examines inputs from the data communications line and the keyboard and places the data to be displayed in a display buffer memory, which is typically a RAM which holds the data for a single or multiple screenload (page) or for a single character row. High-end ('smart' and 'intelligent') terminals start with the same base, but append additional circuits to provide more features and capabilities. The following sections describe the functions of each of the major blocks.

Character Timing and Sync Generation

The major function of this block is to generate the horizontal and vertical timing signals required to produce the TV raster on the CRT monitor. Other functions include the generation of display memory addresses in synchronism with the monitor scan and in accordance with a defined screen format (characters per row, scan lines per row and rows per screen), generation of a cursor signal at the appropriate scan position, and generation of video blanking signals during retrace intervals.

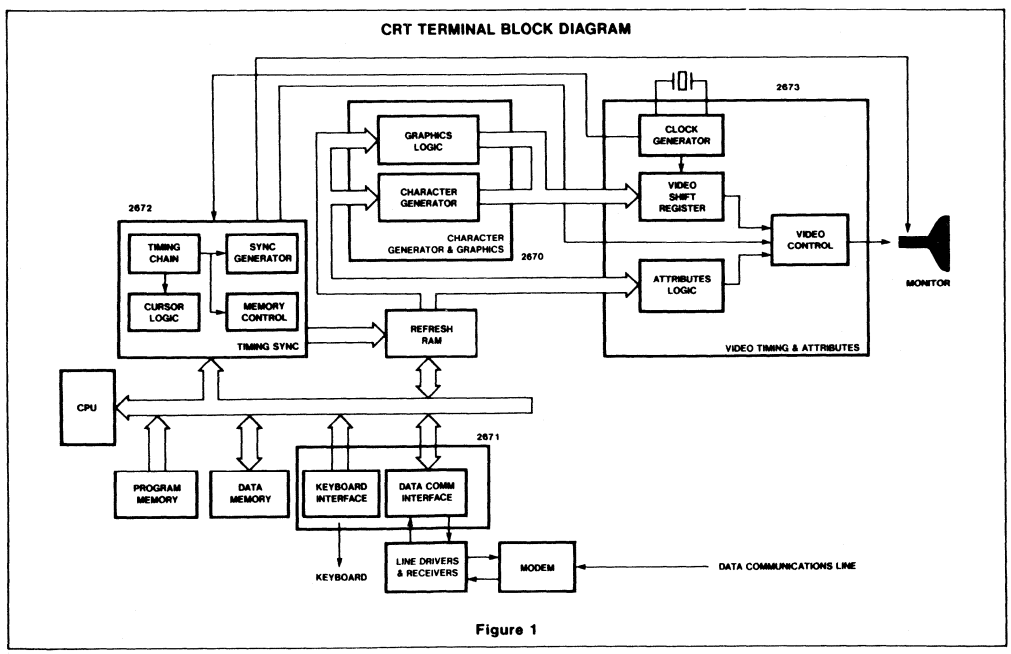


Figure 1

I/O Interface

In its simplest form, this block provides an interface to a keyboard to identify the key depressed and a serial communications link, normally operating in an asynchronous format, between the terminal and the host computer. Although these functions could be performed programmatically by the terminal CPU system, removing these functions to intelligent controllers unburden the system CPU and allow it to be used more effectively to provide additional features with a relatively small cost impact.

Character and Graphics Generation

These circuits convert the data stored in the display memory to the line by line dot patterns required to display the data on the CRT monitor.

Video Timing and Visual Attributes

This section contains the high speed (dot rate) circuits necessary to convert the parallel data from the character and graphics generation circuits to the serial video stream required by the CRT. Also included are circuits to sum visual display attributes such as blinking, high/low intensity, reverse video, and underlining into the video stream.

SIGNETICS' CRT CHIP SET

As mentioned previously, the Signetics CRT 'set' consists of four circuits. The functions of these circuits correspond closely to the four major CRT terminal blocks described above. The circuits have been partitioned so as to allow each to be used independent of the others, allow several alternative methods of implementing the display memory interface so that the hardware can be tailored to the system requirements, provide a full complement of programmable capabilities, and minimize the number of support circuits required.

The following sections give a brief description of each of the circuits. The reader is referred to the individual data sheets for full operational details.

2672 Programmable Video Timing Controller (PVTC)

The 2672 PVTC, figure 2, is a programmable device designed for use in CRT terminals and display systems that employ raster scan techniques. The PVTC generates the vertical and horizontal timing signals necessary for the display of interlaced or non-interlaced data on a CRT monitor. Also, the 2672 provides consecutive addressing to a user specified display buffer memory domain and controls the CPU-display buffer interface for various buffer configuration modes. A variety of operating modes, display formats, and timing profiles can be implemented by programming the control registers in the PVTC.

The CPU initializes the 2672 control and timing registers for the desired timing profiles and memory configuration. The PVTC provides the handshake control for CPU access to the display buffer. One of four memory access modes may be programmed: independent mode, trans-

2672 PROGRAMMABLE VIDEO TIMING CONTROLLER

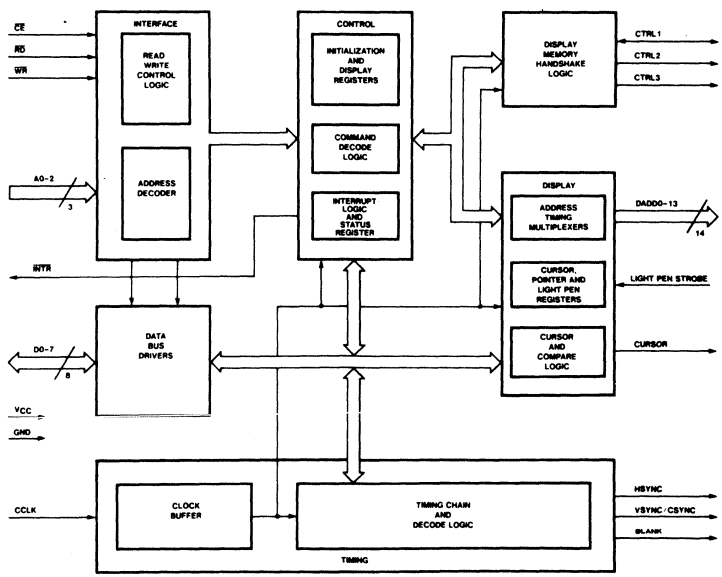


Figure 2

parent mode, shared mode, and row mode. These modes are described in the System Configurations section of this application note.

In all modes, the PVTC provides addresses for the display buffer which outputs the character codes to the 2670 Display Character and Graphics Generator (DCGG) and visual attribute codes to the 2673 Video Attributes Controller (VAC). The DCGG and PVTC supply the dot data and sync timing to the VAC which generates the serialized video.

Programmable features of the PVTC include screen format (characters/row, rows/screen, scan lines/row), horizontal and vertical timing parameters, cursor type (block or underline) and blink rate, character blink rate, interlaced or non-interlaced operation, and single or double height characters.

The PVTC is capable of producing interrupts based upon several internal conditions. By using these interrupts (or by polling the equivalent status register) display features such as non-consecutive buffer addressing for split screen operation, multiple cursors, horizontal and vertical scrolling, and smooth vertical scroll can be implemented.

2671 Programmable Keyboard and Communications Controller (PKCC)

The 2671, figure 3, is an MOS LSI device which provides a versatile keyboard interface and also functions as an asynchronous communications controller. It is intended for use in microprocessor based systems and provides an eight bit data bus interface.

The keyboard controller handles the scanning, debounce, and encoding of mechanical or capacitive keyboards with a maximum of 128 keys utilizing any of four programmable rollover modes. A mask programmable ROM provides four levels of key encoding, corresponding to the separate shift and control input combinations. An eight bit keyboard status register transmits status information to the CPU. Programmable features include rollover mode, scan rate and debounce time, coded or uncoded operation, and automatic repeat operation.

The communications section of the PKCC is a universal asynchronous receiver and transmitter (UART). The receiver accepts serial input data and converts it to parallel data characters. Simultaneously, the transmitter accepts parallel data from the CPU data bus and outputs it in serialized form. Received data is checked for parity and framing errors, and break conditions are flagged. Character lengths can be programmed as 5, 6, 7, or 8 bits not including parity, start or stop bits. An internal baud rate generator (BRG) operating from an external clock or directly from a crystal can be used to derive one of sixteen receive and/or transmit clocks. An eight bit communications status register provides status information to the CPU.

The PKCC has an interrupt mask register to selectively enable keyboard and communications status bits to generate interrupts. Priority encoded interrupt vectoring is available. Upon receipt of an interrupt acknowledge, a mask programmable interrupt vector will be output on the data bus reflecting the source of the interrupt. The mask enabled interrupt sources can also be read directly.

2670 Display Character and Graphics Generator (DCGG)

The DCGG, figure 4, is a mask-programmable 11,648-bit line select character generator. It contains 128 10x9 characters placed in a 10x16 matrix, and has the capability of shifting certain characters, such as j, y, g, p and q, that normally extend below the baseline; effectively, the 9 active lines are lowered within the matrix to compensate for the character's position.

Seven bits of an 8-bit address code are used to select 1 of the 128 available characters. The eighth bit functions as a chip enable signal. Each character is defined by a pattern of logic 1s and 0s stored in a 10x9 matrix. When a specific 4-bit binary line address code is applied, a word of 10 parallel bits appears to the output. The lines can be sequentially selected, providing a 9-word sequence of 10 parallel bits per word for each character selected by the address inputs. As the line address inputs are sequentially addressed, the device will automatically place the 10x9 character in 1 of 2 preprogrammed positions on the 16-line matrix with the positions defined by the 4-line address inputs. One or more of the 10 parallel outputs can be used as control signals to selectively enable functions such as half-dot shift, color selection, etc.

The 2670 DCGG includes latches to store the character address and line address data. A control input to inhibit character data output for certain groups of characters is also provided. The 2670 also includes a graphics capability, wherein the 8-bit character code is translated directly into 256 possible user programmable graphic patterns. Thus, the DCGG can generate data for 384 distinct patterns, of which 128 are defined by the mask programmable ROM.

2673 Video and Attributes Controller (VAC)

The 2673, figure 5, is a bipolar LSI device designed for CRT terminals and display systems that employ raster scan techniques. It contains a high speed video shift register, field and character attributes logic, attribute latch, cursor format logic and half dot shift control, and can be programmed for a light or dark screen background.

The VAC visual attribute capabilities are reverse video, character blank, blink, underline, highlight, and light pen strike-thru or, optionally, graphics. Each attribute has a separate control input which is latched internally when the AFLAG input is asserted. If the AMODE input is low, the attributes are valid for one character time. If AMODE is high, the attributes remain valid until the field is terminated by strobing in a new attributes set. The attributes are double buffered on a row by row basis internally so that field attributes can extend across character row boundaries thereby eliminating the necessity of starting each row with an attribute set.

The horizontal dot frequency is the basic timing input element to the VAC; internally, this clock is divided down to provide a character clock output for system synchronization. Ten bits of dot data are parallel loaded into the video shift register on each character boundary. The video data is shifted out on three outputs at the dot frequency. On the video output, the video is presented as a three level signal representing low, medium and high intensities, and the three intensities are also encoded on the two TTL compatible video outputs.

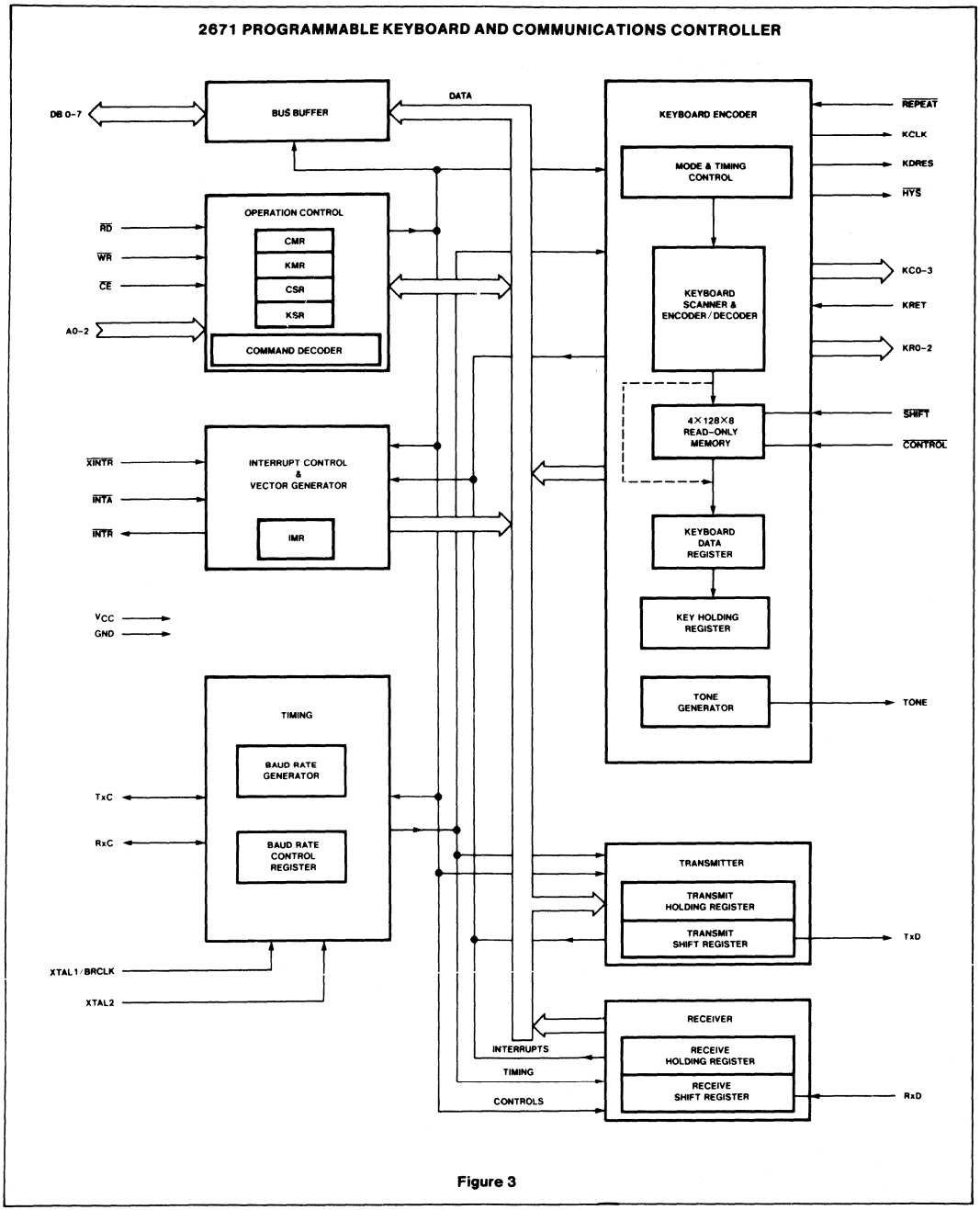

2671 PROGRAMMABLE KEYBOARD AND COMMUNICATIONS CONTROLLER
DB 0-7
BUS BUFFER
DATA
KEYBOARD ENCODER
REPEAT
KCLK
KDRES
HYS
MODE & TIMING CONTROL
OPERATION CONTROL
RD
WR
CE
A0-2
CMR
KMR
CSR
KSR
COMMAND DECODER
KEYBOARD SCANNER & ENCODER/DECODER
KC0-3
KRET
KR0-2
4X128X8 READ-ONLY MEMORY
SHIFT
CONTROL
XINTR
INTA
INTR
INTERRUPT CONTROL & VECTOR GENERATOR
IMR
KEYBOARD DATA REGISTER
KEY HOLDING REGISTER
VCC
GND
TONE GENERATOR
TONE
TIMING
BAUD RATE GENERATOR
TxC
RxC
BAUD RATE CONTROL REGISTER
TRANSMITTER
TRANSMIT HOLDING REGISTER
TRANSMIT SHIFT REGISTER
TxD
XTAL1/BRCLK
XTAL2
RECEIVER
INTERRUPTS
RECEIVE HOLDING REGISTER
TIMING
RECEIVE SHIFT REGISTER
CONTROLS
RxD

Figure 3

2670 DISPLAY CHARACTER AND GRAPHICS GENERATOR

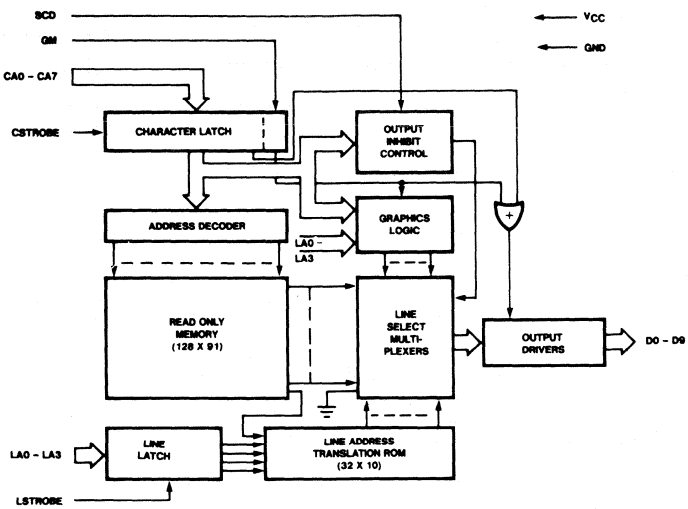


Figure 4

2673 VIDEO AND ATTRIBUTES CONTROLLER

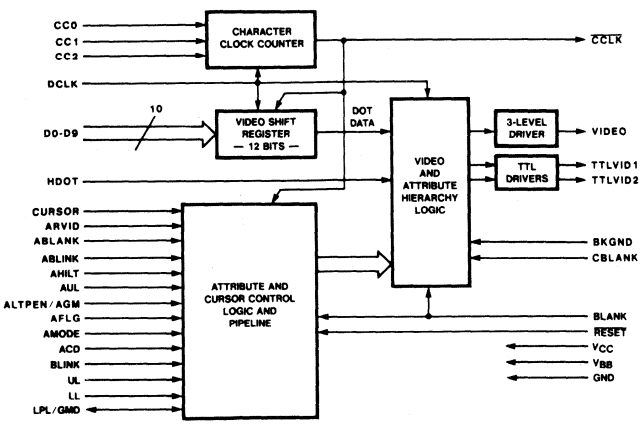


Figure 5

SYSTEM CONFIGURATIONS

The PVTC supports four common system configurations of display buffer memory interface, designated the independent, transparent, shared, and row buffer modes. The first three modes utilize a single or multiple page RAM and differ primarily in the means used to transfer display data between the RAM and the CPU. The row buffer mode makes use of a single row buffer (which can be a shift register or a small RAM) that is updated in real time to contain the appropriate display data.

Independent Mode

The CPU to RAM interface configuration for this mode is illustrated in figure 6. Transfer of data between the CPU and display memory is accomplished via a bidirectional latched port and is controlled by the PVTC signals read data buffer ($\overline{RDB}$), write data buffer ($\overline{WDB}$), and buffer chip enable ($\overline{BCE}$). This mode provides a non-contention type of operation that does not require address multiplexers. The CPU does not address the memory directly - the read or write operation is performed at the address contained in the cursor address register or the pointer address register as specified by the CPU. The PVTC enacts the data transfers during blanking intervals in order to prevent visual disturbances of the displayed data.

The CPU manages the data transfers by supplying commands to the PVTC. The commands used are:

1. Read/Write at pointer address.
2. Read/Write at cursor address (with optional increment of address).
3. Write from cursor address to pointer address.

The operational sequence for a write to memory operation is:

1. The CPU loads data to be written into the display memory into the interface latch.
2. The CPU writes the destination address into the PVTC's cursor or pointer registers.
3. The CPU checks the PVTC 'RDFLG' status bit to assure that any previous operation has been completed.
4. The CPU issues a 'write at cursor with/without increment' or a 'write at pointer' command to the PVTC.
5. The PVTC negates 'RDFLG', outputs the specified address, and generates control signals to perform requested operation. Data is copied from the interface latch into the memory.
6. The PVTC sets its 'RDFLG' status to indicate that the write operation is completed.

Similarly, a read operation proceeds as follows:

1. Steps 2 and 3 as above.
2. The CPU issues a 'read at cursor with/without increment' or 'read at pointer' command.
3. The PVTC negates 'RDFLG', outputs the specified address, and generates control signals to perform the read operation. Data is copied from the memory to the interface latch and the PVTC sets its 'RDFLG' status to indicate that the operation is completed.
4. The CPU checks the 'RDFLG' status to see if the read is completed.
5. The CPU reads the data from the interface latch.

Loading the same data into a block of display memory is accomplished via the 'write from cursor to pointer' command:

1. The CPU loads the data to be written into the display memory into the interface latch.
2. The CPU writes the beginning address of the memory block into the PVTC's cursor address register and the ending address of the block into the pointer address register.
3. The CPU checks the 'RDFLG' status bit to assure that any previous operation has been completed.
4. The CPU issues a 'write from cursor to pointer' command to the PVTC.

INDEPENDENT BUFFER MODE CONFIGURATION

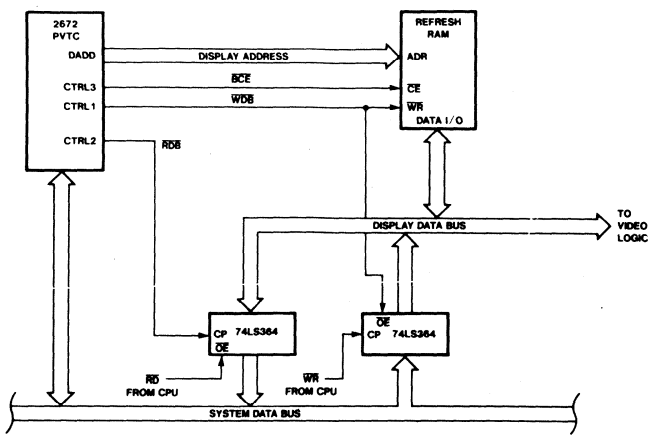


Figure 6

5. The PVTC negates 'RDFLG' and outputs block addresses and control signals to copy the data from the interface latch into the specified block of memory.
6. The PVTC sets its 'RDFLG' status to indicate that the block write is completed.

Similar sequences can be implemented on an interrupt driven basis using the READY interrupt output from the PVTC to inform the CPU that a previously requested command has been completed.

Two timing sequences are possible for the 'read/write at cursor/pointer' commands. If the command is given during the active display window (defined as first scan line of the first character row to the last scan line of the last character row), the operation takes place during the next horizontal blanking interval. If the command is given during the vertical blanking interval, or while the display has been commanded blanked, the operation takes place immediately.

For the 'write from cursor to pointer' operation, the PVTC's BLANK output is asserted automatically and remains asserted until the vertical retrace interval following completion of the command. The memory is filled at a rate of one location per two character times, plus a small amount of overhead.

Shared and Transparent Buffer Modes

In these modes the display buffer RAM is a part of the CPU memory domain and is addressed directly by the CPU. Both modes use the same hardware configuration with the CPU accessing the display buffer via three-state drivers (see figure 7). The processor bus request ($\overline{PBREQ}$) control signal informs the PVTC that the CPU is requesting access to the display buffer. In response to this request, the PVTC raises bus acknowledge ($\overline{BACK}$) until its bus external ($\overline{BEXT}$) output has freed the display address and data busses for CPU access. $\overline{BACK}$, which can be used as a 'hold' input to the CPU, is then lowered to indicate that the CPU can access the buffer.

In transparent mode, the PVTC delays the granting of the buffer to the CPU until a vertical or horizontal blanking interval, thereby causing minimum disturbance of the display. In shared mode, the PVTC will blank the display and grant immediate access to the CPU.

Row Buffer Mode

Figure 8 shows the hardware implementation for the row buffer mode. During the first scan line (line 0) of each character row, the PVTC halts the CPU and DMA's the next row of character data from the system memory to the row buffer memory. The PVTC then releases the CPU and displays the row buffer data for the programmed number of scan lines. The bus request ($\overline{BREQ}$) control signal informs the CPU that character addresses and the memory bus control (MBC) signal will start at the next falling edge of BLANK. The CPU must release the address and data busses before this time to prevent bus

PVTC SHARED OR TRANSPARENT BUFFER MODES

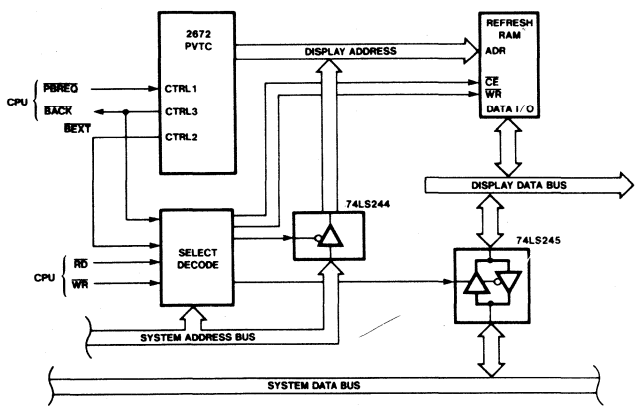


Figure 7

ROW BUFFER MODE CONFIGURATION

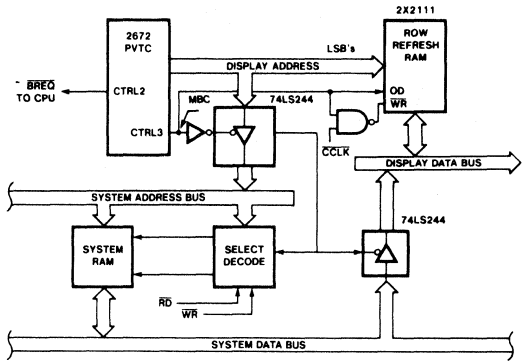


Figure 8

contention. After the row of character data is transferred to the CPU, BREQ returns high to grant memory control back to the CPU.

A MINIMUM CHIP COUNT TERMINAL IMPLEMENTATION

Figure 9 is the schematic of a minimum chip count CRT terminal using the four CRT set devices. Only 15 IC packages are required for the complete implementation, including all keyboard encoding and RS-232 level conversion for the serial interface. Despite this low chip count the terminal is capable of providing an impressive array of features including:

Display Format:
- 24 or 25 character rows
- 80 characters per row

Character Format:
- 7x9 dot matrix character in a 9x12 character block
- 96 ASCII alphanumeric characters
- 32 special symbols
- Block graphics
- Line drawing character set

Cursor:
- Underline or block cursor
- Optional blinking

Keyboard:
- 128 keys maximum
- Non-encoded
- Cursor control keys
- Numeric keypad

Serial Interface:
- Full or half duplex
- RS-232 compatible
- 16 baud rates with internal baud rate generator
- Character or block transmission

Operating Modes:
- Normal
- Transparent (displays graphic and control characters)
- Page or scroll with optional smooth scroll

Visual Attributes:
- Blink
- Reverse video
- Highlight
- Underline
- Non-display

The system utilizes the independent buffer mode to minimize hardware requirements. The dual port interface to the 2Kx8 display buffer is via a Signetics 8X31 bidirectional latch. This may be replaced by a unidirectional latch such as the 74LS374 if reading of the RAM's contents by the CPU is not required.

The operating program for the terminal is contained in the internal ROM of the 8049 microcomputer, which also provides the RAM required by the system program. Since the majority of the terminal's features are tailored by firmware, the ROM size can be increased, either internally or externally, to support additional functions.[1]

BASIC TERMINAL SOFTWARE

The software for a microcomputer based terminal is closely tied to the system hardware configuration and its characteristics. If an interrupt driven mode of operation is desired, the system hardware/software design must be capable of prioritizing the interrupts so that the system will correctly service interrupts from different sources. In a typical system, there are three interrupt sources: the keyboard, the communications interface, and the video timing controller. The latter must usually be assigned the highest priority since failure to service an interrupt from the video timing controller on a timely basis may result in visual perturbations on the display. The keyboard and datacomm interrupts can, in most cases, absorb some time delay before they are serviced since they include one or more levels of data buffers.

[1]A pre-programmed 8049 microcomputer containing the operating firmware for this terminal will be available from Signetics.

MINIMUM CHIP COUNT TERMINAL

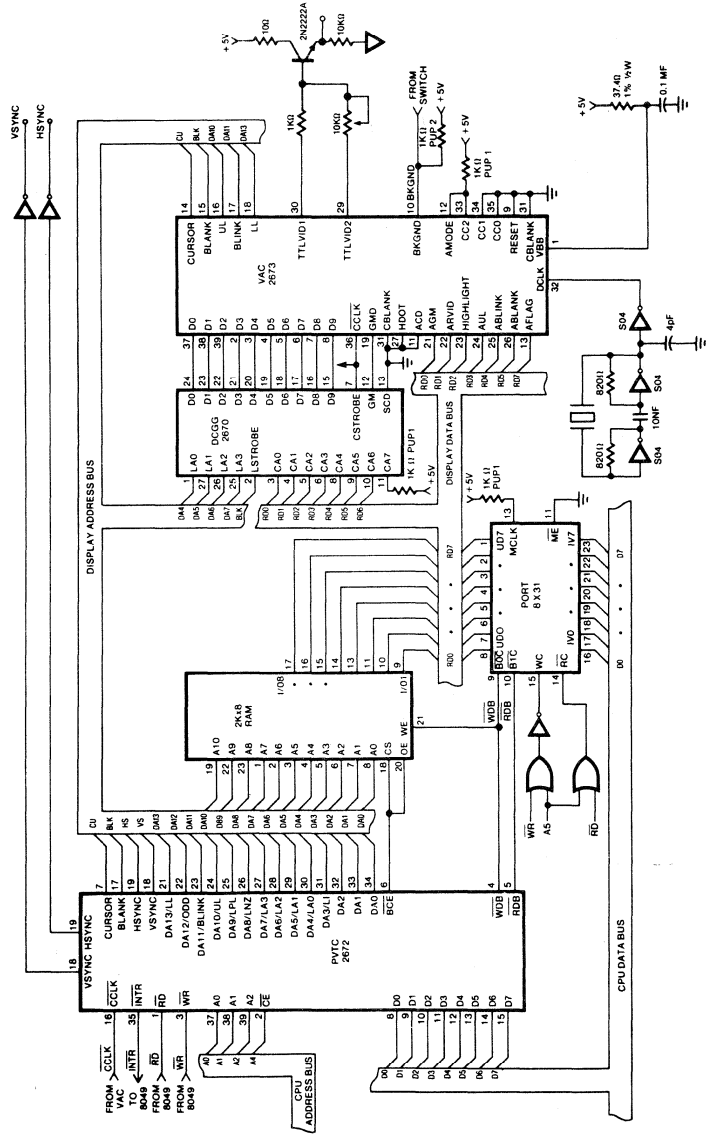


Figure 9 (Continued)

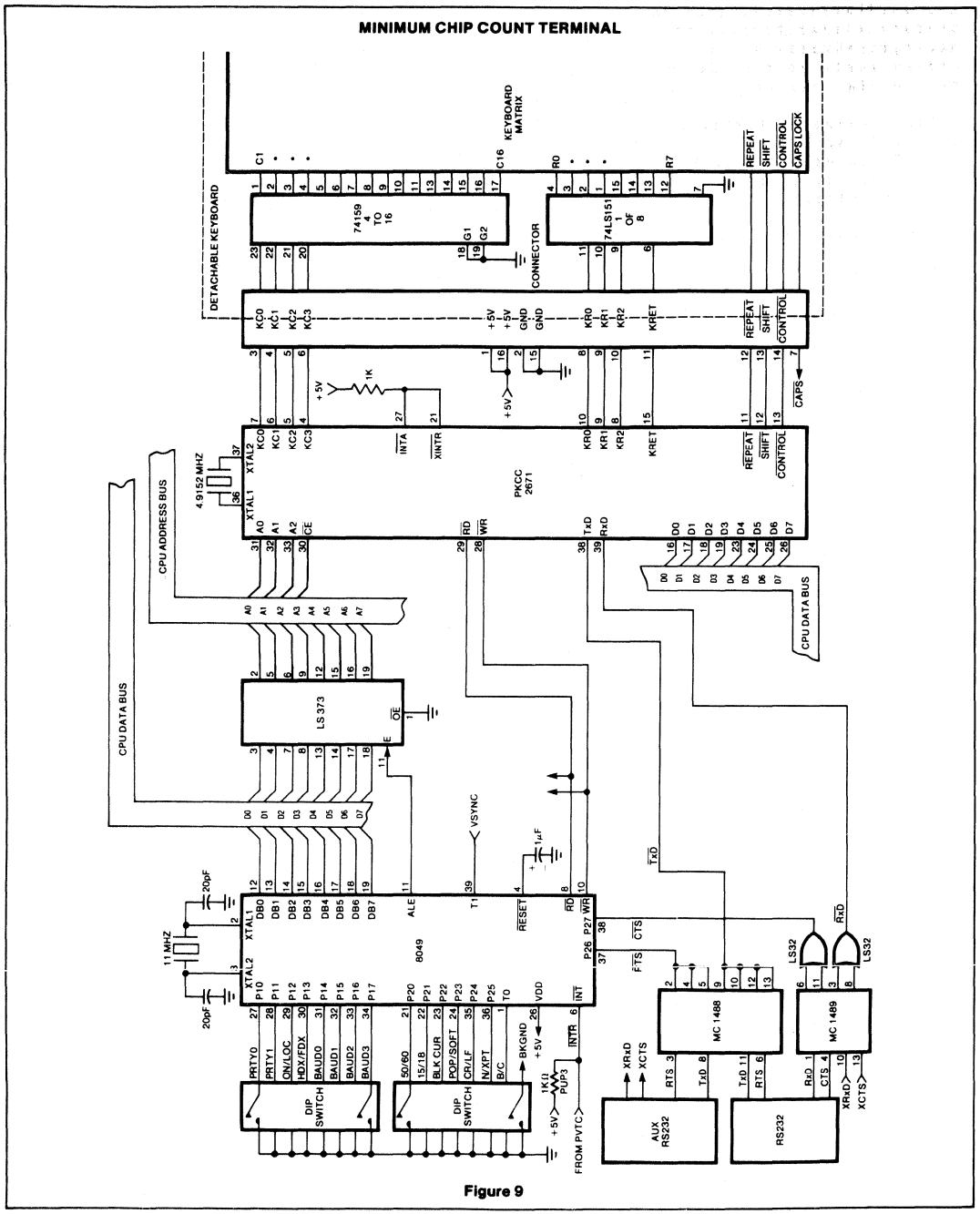

MINIMUM CHIP COUNT TERMINAL
DETACHABLE KEYBOARD
KEYBOARD MATRIX
CONNECTOR
74159 4 TO 16
74LS151 1 OF 8
PKCC 2671
8049
LS 373
CPU ADDRESS BUS
CPU DATA BUS
4.9152MHZ
11 MHZ
20pF
1µF
VSYNC
DIP SWITCH
AUX RS232
RS232
MC 1488
MC 1489
LS32
FROM PVTC

Figure 9

Often, a multi-level interrupt structure will be required so that a high priority interrupt requiring immediate service can be serviced even while the system is in the process of servicing a lower priority interrupt.

A simplified flowchart for the software for an interrupt driven terminal is shown in figure 10. After application of power, the microprocessor first performs a system initialization routine which consists of five parts:

1. Clear the microcomputer's scratchpad RAM.
2. Initialize the 2672 PVTC for the desired screen format, monitor timing parameters, cursor parameters, and display start address.
3. Clear the CRT display by loading a non display-code (usually an ASCII 'space', 20 hex) into the buffer memory.
4. Initialize the 2671 PKCC for the desired keyboard and serial interface modes.
5. Read any mode switches (e.g., full or half duplex, baud rate, cursor type, etc.) and set system parameters as required.

The processor can now enable its interrupts and wait in a loop until an interrupt is received. When this happens, the processor first determines the source of the interrupt and then performs the required system operation.

An interrupt from the CRT timing controller usually indicates that some information is required for proper screen refresh operation. For example, the PVTC may issue a 'split screen' interrupt to indicate that a new address must be loaded into its screen start registers in order for the next character row to be displayed from other than the next sequential address in memory. The CPU must service this interrupt within a finite time in order for the display to operate correctly.

An interrupt from the keyboard interface may be either a displayable character or a control function. Displayable characters are usually transmitted to the host computer and also placed into the buffer memory for display on the terminal. Certain control characters, such as cursor control keys or keyboard error codes, may cause only local actions, while others will also require transmission to the host.

An interrupt from the data communications interface may also be a displayable character or a system control character. In either case the microprocessor must determine the type of character and perform the necessary system operation.

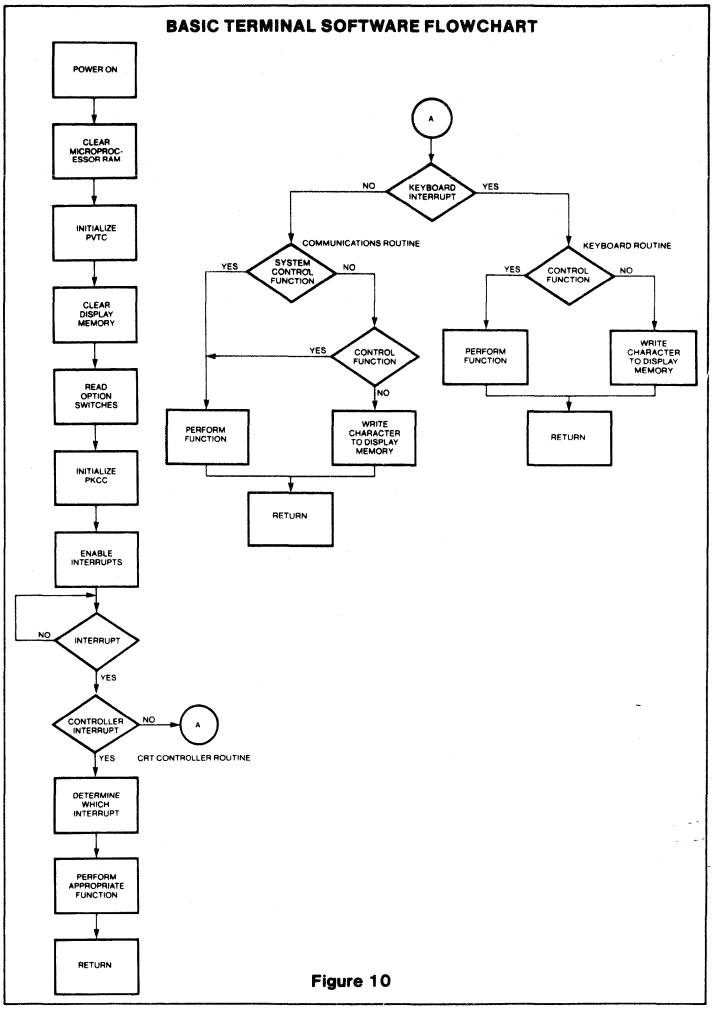


Figure 10

A DESIGN EXAMPLE

A fully operational emulation of an IBM 3101 terminal was designed and constructed using the Signetics CRT chip set. The terminal incorporates the majority of the 3101's functions. Selected functions were not incorporated due to program memory limitations. For example, the tabbing functions were developed and tested but were left out in deference to the block transmission functions. More features could have been included by selecting another of the numerous microprocessor devices on the market with greater program memory capacity. Major features of the terminal are summarized in table 1.[1]

[1]A data package for the design, including details of operation, schematic, and program listing, is available upon request by writing to:
Signetics Corporation
Microprocessor Applications Dept.
Mail Station 12-76
P.O. Box 409
Sunnyvale, CA 94086

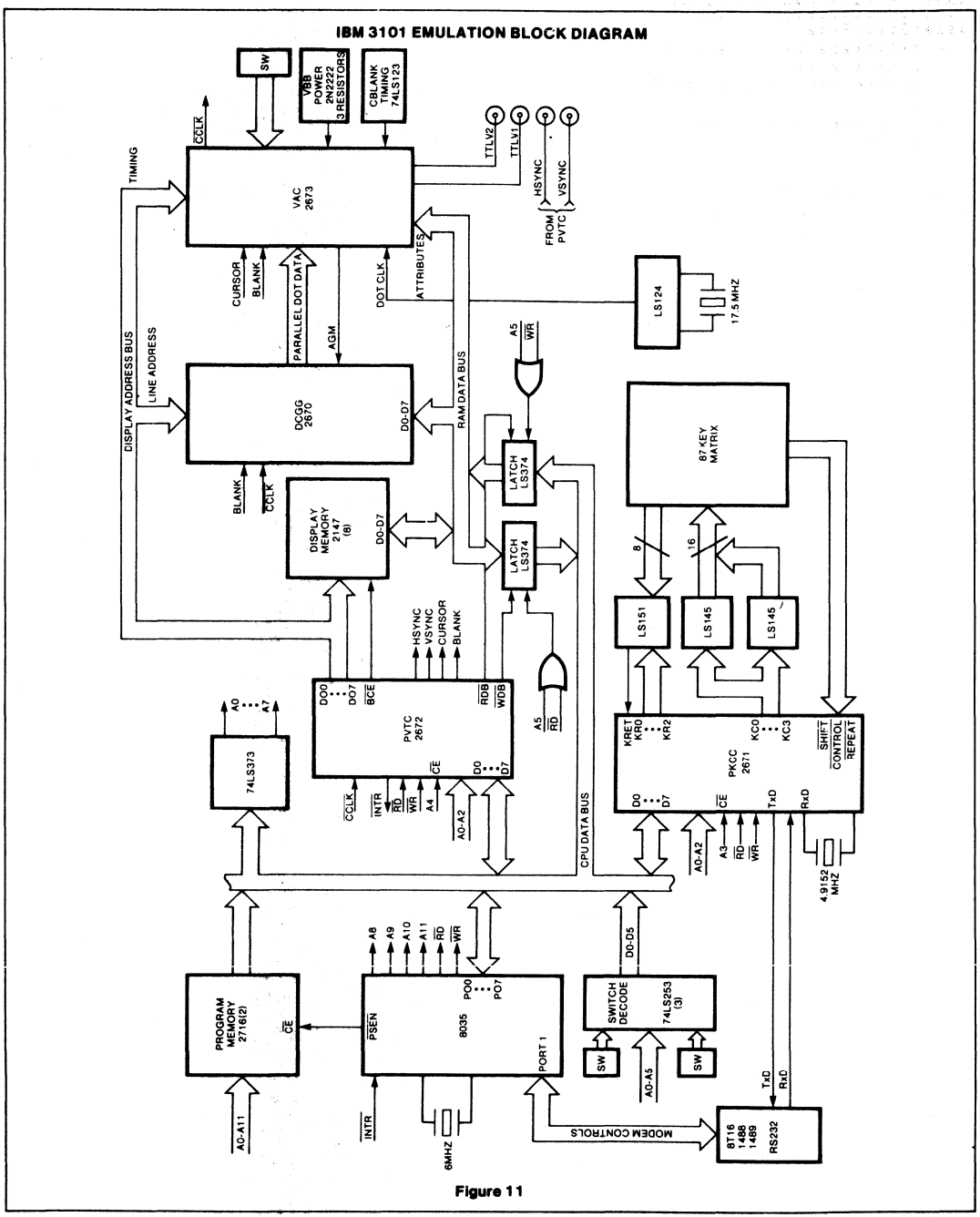

IBM 3101 EMULATION BLOCK DIAGRAM
SW
VBB POWER 2N2222 3 RESISTORS
CBLANK TIMING 74LS123
CCLK
TIMING
VAC 2673
TTLV2
TTLV1
HSYNC
VSYNC
FROM PVTC
CURSOR
BLANK
PARALLEL DOT DATA
AGM
DOT CLK
ATTRIBUTES
LS124
17.5 MHZ
A5
WR
DISPLAY ADDRESS BUS
LINE ADDRESS
DCGG 2670
D0-D7
RAM DATA BUS
LATCH LS374
87 KEY MATRIX
BLANK
CCLK
DISPLAY MEMORY 2147 (8)
D0-D7
LATCH LS374
8
16
LS151
LS145
LS145
HSYNC
VSYNC
CURSOR
BLANK
A0 ... A7
74LS373
D00 ... D07
BCE
PVTC 2672
RDB
WDB
A5
RD
KRET
KR0 ... KR2
KC0 ... KC3
SHIFT CONTROL
REPEAT
PKCC 2671
D0 ... D7
CE
TxD
RxD
CCLK
INTR
RD
WR
A4
A0-A2
D0 ... D7
CE
CPU DATA BUS
A0-A2
A3
RD
WR
4.9152 MHZ
A8
A9
A10
A11
RD
WR
P00 ... P07
D0-D5
PROGRAM MEMORY 2716(2)
CE
PSEN
8035
PORT 1
SWITCH DECODE
74LS253 (3)
SW
SW
A0-A5
TxD
RxD
A0-A11
INTR
6MHZ
MODEM CONTROLS
8T16 1488 1489
RS232

Figure 11

Table 1 — TERMINAL FEATURES

Display Screen Format
- 2000 character screen capacity (25 rows x 80 columns)
- Operator information area (25th line)
- Block-shaped cursor with optional blinking

Displayable Graphic Set
- 95 ASCII characters for non-transparent mode
- 128 characters for transparent mode
- 7x9 character matrix in 9x12 field

Keyboard
- 63-key main keyboard
- 12-key control key cluster
- 12-key numeric keypad
- Keyboard lock/unlock under software control
- Keyboard clicker
- Typamatic operation

Edit Functions
- Cursor controls: up, down, left, right, home
- Cursor address read and write
- Erase functions: erase EOL, erase EOS, clear screen

Visual Attributes
- Highlighted field
- Blinking field
- Non-displayed field
- Underlined field

Modes of Operation
- Transmission modes: character or block (page or line)
- Normal or transparent

Line Protocol
- Asynchronous
- 7-bit ASCII with programmable parity
- One or two stop bits
- Full or half duplex
- Online or local
- Programmable line turnaround character for block mode (EOT/ETX/CR/XOFF)
- EIA RS232 interface
- Communication line speed: 50 to 9,600 baud

Screen Refresh Rate
- 60 Hz

Terminal Hardware

The block diagram of the 8035 based terminal is illustrated in figure 11. It is an expanded version of the logic shown in figure 9, the major difference being a larger display RAM, to provide up to two pages of screen data, and the addition of several input ports to handle the large number of option and set-up switches. The terminal's software is contained in 4K of program storage external to the 8035.

The 2672 PVTC is programmed to operate in the independent buffer mode with the CPU isolated from the display RAM by two 74LS374 eight-bit latches, which provide the path for data transfers between the CPU and RAM. The PVTC, responding to commands from the CPU, completely controls the data transfer. To avoid display interference, the PVTC is instructed to complete the access during a blanking interval. For massive display updates (clear screen, load form, etc.) the PVTC is instructed to blank the display and service the data transfer immediately and continuously. Additional memory contention circuitry is not necessary since the PVTC provides all of the timing and addressing (via cursor and pointer) necessary to complete the transfer. An interrupt from the PVTC informs the CPU when an operation is completed.

The PVTC addresses the display buffer memory, which contains both character and attribute data. An attribute byte is identified by the software by setting bit 7 of the byte to a logic 1. The RAM data outputs are applied to the 2670 DCGG, which provides the character dot data information, and to the 2673 VAC.

The VAC is hardwired to operate in the field attributes mode for this application. An attribute character occupies a screen position but is not displayed unless the ACD input to the VAC is asserted. Bit 7 of the character byte identifies a character as an attribute character if it is a 1. When bit 7 on the RAM data bus is a 1, the attribute byte is latched into the VAC to begin a new attributes field. Since the attributes are double buffered in the VAC, only one byte (at any character position) is required to specify a field.

The bipolar VAC circuit serializes the dot data from the DCGG into a 17.5 MHz data stream for the monitor. Two TTL-level video outputs provide three levels of video: black, white, and gray.

The PKCC provides the asynchronous data communications link at one of sixteen selectable baud rates. The PKCC addresses two 74LS145s which act as a 4-to-16 decoder to drive a 16x8 matrix keyboard. Key depressions are detected on the KRET input from a 74LS151 8-to-1 multiplexer. Each key depression is debounced, encoded according to the states of the SHIFT and CONTROL inputs, and presented to the CPU. Repeat and 'typomatic' (auto-repeat) functions are processed automatically by the PKCC.

Timing Calculations

One of the tasks required in the design phase of the terminal is the selection of a suitable monitor and calculation of the PVTC register values to provide suitable drive signals for the selected monitor.

The selection process begins with calculation of the required horizontal scan frequency. Each character will be contained in a 9 dot by 12 line field. Since there are 25 display rows, the total number of active scan lines will be 12 x 25, or 300. To this we must add some number of scan lines for the vertical retrace, which is typically 5 to 10 percent of the active scan lines. For a screen refresh rate of 60 Hz, this yields

H frequency = (60)(300)(1.1) = 18,900 Hz.

A Motorola monitor was selected for the application. The major timing specifications for the monitor are:

Horizontal frequency: 18.72 KHz ± 500 Hz
Horizontal retrace: 8 us max
Horizontal sync width: 4 us min
Vertical frequency: 50/60 Hz
Vertical retrace: 750 us max
Vertical sync width: 50 us min

Monitor timing definitions are shown in figure 12. The worksheet illustrated in table 2 can be used to compute the required timing and associated PVTC register values. Some rough guesses are required initially and several iterations through the worksheet will usually be required to arrive at final values. For example, the character clock period must be known to select the horizontal front porch (HFP), sync width (HSYNC), and back porch (HBP) values. An estimate of the character period can be made initially as follows:

Horizontal period = 1/18,900 = 52.9 us
Horizontal active = total - blank = 52.9 — 10 = 42.9 us
Character period = 42.9/80 = 0.53 us approximately

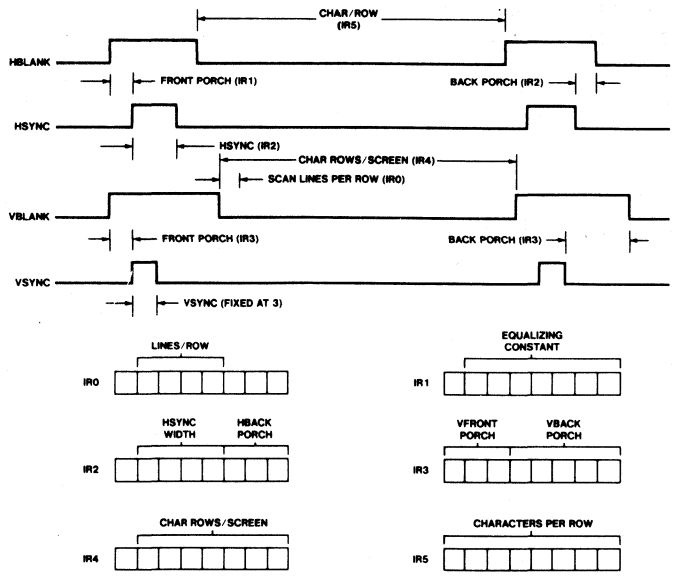


Figure 12

In calculating horizontal timing, an approximate ratio for the HFP, HSYNC, and HBP of 1:2:2 respectively is recommended.

Table 2 contains the final values selected for the application.

Memory Allocation

The 4K bytes of available buffer memory were allocated as follows (all addresses are in hex):

- 0000 to 004F: display data for row 25, status line
- 0050 to 0075: not used
- 0076 to 007F: CPU scratchpad
- 0080 to 07FF: display data for rows 1 to 24
- 0800 to 0FFF: not used, available for second page of display data

The PVTC's 'display buffer first address' and 'display buffer last address' registers are loaded with the values 0080 and 07FF respectively so as to cause this portion of the RAM to act as a circular buffer. Initially the display data is organized in the RAM as follows:

- 0080 to 00CF: row 1 data
- 00D0 to 011F: row 2 data

  ⋮
- 07B0 to 07FF: row 24 data

When a scroll operation is required, the CPU changes the value in the PVTC's 'screen start' register from 0080 to 00D0. This effectively shifts the displayed data up one row. Upon reaching the specified last buffer address (which is now the last character in row 23), the PVTC automatically changes the addressing sequence to resume starting at 0080 for the 24th row. The display data is now organized:

- 00D0 to 011F: row 1 data
- 0120 to 016F: row 2 data

  ⋮
- 07B0 to 07FF: row 23 data
- 0080 to 00CF: row 24 data

The CPU can clear the previous data in 0080 to 00CF so that a blank row appears in the 24th position.

The status line (row 25) data is kept in a separate section of RAM to eliminate the necessity of moving the data whenever the scrolling operation described above occurs. Thus, the PVTC must be instructed to change its addressing sequence at the beginning of the 25th row. This is accomplished by use of the split screen row interrupt capability. IR10, the 'split screen interrupt row' register, is initialized so as to cause an interrupt to be issued at the beginning of row 24. The CPU responds to this interrupt by changing the value in the screen start register to 0000. The PVTC then uses this value as the starting address of the next (25th) row, causing the status line to be displayed in that position. The CPU must re-load the screen start register before the end of the vertical blanking interval with the correct value for the first character to be displayed on the screen.

Terminal Software

Because the 8035 microcomputer used in the terminal provides only a single interrupt level, a totally interrupt driven software design could not be used. The interrupt was assigned to the PVTC to service the split screen interrupt described above and the operations required to implement the smooth scroll feature. The keyboard and datacomm functions are serviced by polling the PKCC status register. Both the keyboard interface and UART receiver are double buffered in the PKCC, preventing overrun even if they are not serviced immediately.

The program generally follows the typical program flow described previously. At system reset the 8035 interrupts are dis-

Table 2 — CRT TIMING WORKSHEET

1.	HORIZONTAL CHARACTER BLOCK (no. of dots)	9	
2.	VERTICAL CHARACTER BLOCK (no. of scan lines)	12	(IR0)
3.	VERTICAL REFRESH RATE, Hz	60	
4.	CHARACTERS PER ROW	80	(IR5)
5.	CHARACTER ROWS PER SCREEN	25	(IR4)
6.	TOTAL ACTIVE VIDEO SCAN LINES (step 2 x step 5)	300	
7.	VERTICAL FRONT PORCH (no. of scan lines)	4	(IR3)
8.	VERTICAL BACK PORCH (no. of scan lines)	12	(IR3)
9.	VERTICAL RETRACE INTERVAL (step 8 + 3)	15	
10.	TOTAL SCAN LINES PER FRAME (add steps 6, 7, and 9)	319	
11.	HORIZONTAL LINE RATE, KHz (step 3 x step 10)	19.14	
12.	HORIZONTAL FRONT PORCH (character time units)	5	
13.	HORIZONTAL SYNC WIDTH (character time units)	8	(IR2)
14.	HORIZONTAL BACK PORCH (character time units)	9	(IR2)
15.	HORIZONTAL RETRACE INTERVAL (step 13 + step 14)	17	
16.	TOTAL CHARACTER TIME UNITS IN ONE HORIZONTAL SCAN LINE (add steps 4, 12, 13, and 14)	102	
17.	EQUALIZING CONSTANT ([step 16 / 2] — [2 x step 13])	35	(IR1)
18.	CHARACTER CLOCK RATE, MHz (step 16 x step 11)	1.95228	
19.	CHARACTER PERIOD, us (1 / step 18)	0.512	
20.	SCAN LINE PERIOD, us (step 19 x step 16)	53.27	
21.	DOT CLOCK RATE, MHz (step 18 x step 1)	17.57052	

	PARAMETER	SPEC	ACTUAL
A.	HORIZONTAL RATE, KHz	18.72 ± 0.5	19.14
B.	HORIZONTAL RETRACE TIME, us	8	8.7
C.	HORIZONTAL SYNC WIDTH	4	4.1
D.	VERTICAL RATE, Hz	50 - 60	60
E.	VERTICAL RETRACE TIME, us	750	784
F.	VERTICAL SYNC WIDTH, us	50	157

abled, data memory and display memory are cleared to zeroes, and both the PVTC and PKCC are master reset through software commands. The system option switches are then read and stored and the PVTC and PKCC internal registers are initilized for the selected operation. Finally, the initial data for the status line is loaded, the PVTC, UART, and keyboard are enabled, and the CPU interrupt is enabled.

The program then enters a loop where the PKCC is checked for keyboard or UART entries. If an entry has occurred, the character is fetched and stored in a software controlled FIFO (first-in-first-out) memory which is eight bytes deep for both receiving or transmitting characters (the need for the FIFO is described below). If either FIFO has an entry, the program proceeds to a character recognition routine which checks for the type of character (displayable or control) and the appropriate handling subroutine (ESC sequence, control sequence, cursor control, character display, etc.) is called. If the FIFO's are empty, the polling routine checks the option switches for any changes since reset entry and if so reconfigures the system as necessary.

The need from the FIFOs results from the method used to effect the clear row function required when a scroll is performed. Although the PVTC includes a 'clear from cursor to pointer' command that can be used to clear a block of memory rapidly, the display is temporarily blanked during this operation. This would cause undesirable flashes on the display. Instead, the program does the function by a repetitive loop using the 'write at cursor and increment' command. Since the write occurs only once per scan line during the active display window, a worst case total of approximately 80 scan line times is required to execute the routine. This would limit the maximum received character rate to approximately one per 80 scan lines or about 240 characters per second (2400 baud). To overcome this limitation, the PKCC is also polled each time through the clear line subroutine loop, and any entries from the receiver or keyboard are stored in the appropriate FIFO. Since the FIFO is eight deep, this allows eight characters to be received in the same time, increasing the maximum baud rate to 19,200. (Other program limitations actually reduce the maximum baud rate to 9600 baud). However, this does not increase the rate at which characters which cause a scroll function to occur, such as a line feed, can be received. Each character of this type must be followed by 'fill' characters in order for data rates higher than 2400 baud to be used.

An interrupt from the PVTC will occur when the display scan reaches the row count programmed in its split screen address register, row address 24 (for the 24th row). In response to the interrupt, the CPU loads the screen start registers with the address of the status line (0000) and enables the PVTC's line zero interrupt. This causes another interrupt at the beginning of display of the status line. At this time the CPU reloads the screen start register with the proper address to begin the next display frame and disables the line zero interrupt.

If scrolling is required the screen start register value is incremented by 80 (popping off the top row) and the effective bottom row cleared to nulls. If soft scrolling is

selected, additional functions are performed during the interrupt routines. To begin the operation, the line zero interrupt routine adds ten lines to the vertical back porch. This causes the next active screen display to begin ten scan lines later than normal and gives the effect of the display moving up two scan lines (12 lines per character row - 10) instead of jumping up 12 lines. If nothing else were changed, however, the bottom of the display would move down ten lines. Thus, during the row 24 interrupt the number of scan lines per character row is changed to two (12-10), causing only the first two scan lines of that row to be shown. The next line zero interrupt (at row 25) restores the lines per row count back to 12 to keep the whole status line showing, and now changes the vertical back porch to 8. The display moves up two more scan lines and at the next row 24 interrupt four scan lines are shown. The process continues in this manner, providing the effect of the entire display, except for the status line, smoothly scrolling up over a selected interval of six frames, or one tenth of a second.

2670/71/72/73 CRT SET APPLICATION BRIEFS

INTRODUCTION

The Signetics CRT chip set consists of four LSI devices which, when combined with standard microcomputer, memory, and TTL products, permit the implementation of a CRT terminal in as few as 15 total packages. The four LSI devices are:

2670 Display Character and Graphics Generator (DCGG)

The 2670 is a mask programmable line select character generator which contains the dot patterns for 128 10x9 characters. It also provides a semi-graphics capability wherein the 8-bit character code is translated directly into 256 graphic patterns useful for presenting data such as graphs and forms on the CRT display. Additional features of the DCGG include character and line address latches and internal descend logic.

2671 Programmable Keyboard and Communications Controller (PKCC)

The 2671 provides a versatile keyboard interface and an asynchronous communications interface in a single package. The keyboard section handles the scanning, debounce, and encoding of mechanical or capacitive keyboards with up to 128 keys utilizing any of four programmable rollover modes. An internal ROM provides any of four key codes for a depressed key. The communications section is a universal asynchronous receiver and transmitter (UART) with programmable character length, parity, and stop bits. A baud rate generator providing 16 standard communications frequencies which operates directly from a crystal is also incorporated in the 2671.

2672 Programmable Video Timing Controller (PVTC)

The 2672 is designed for use in CRT terminals and other display systems that employ raster scan techniques. It generates the vertical and horizontal timing for the CRT monitor, and provides the addressing for the display buffer memory. The CPU to display buffer interface can be programmed for several different modes of operation, including full screen buffer, multiple page buffer, or row buffer, as required by the application. Programmable features of the PVTC include screen format (characters per row, scan lines per row, rows per screen), horizontal and vertical timing parameters, cursor type, interlaced or non-interlaced operation, and character and cursor blink timing.

2673 Video and Attributes Controller

The 2673 is a bipolar LSI device that contains the high speed timing circuits required in CRT terminal systems. Included on the 2673 are a dot clock counter, video shift register, field and character attributes logic, and cursor display circuits. The 2673 attributes capabilities are reverse video, character blank, blink, underline, highlight, light pen, and graphics. The device provides both TTL and analog video outputs and operates at dot frequencies up to 25MHz.

Individual data sheets are available which describe each of the devices in full detail. A previously published application note entitled "Using the 2670/71/72/73 CRT Terminal Chip Set" (App Note 401), describes the implementation of CRT terminals using the chip set. The purpose of this application note is to provide information on the implementation of special end-product features.

SMOOTH (SOFT) SCROLLING

Scrolling is used in CRT terminals to provide the effect of an 'endless page' on which the data can be written. In normal implementations, once the screen fills up, the space for the next row of data is provided by removing the top row and moving all the remaining rows up by one row, thus creating a blank data row at the bottom of the screen into which the new information is placed. (The process may be reversed if the new data is to be written at the top of the screen). This technique creates a readability problem when viewing data which is being received at a relatively high speed, since the rows are jumping up (or down) at a fast rate. Smooth or soft scrolling improves readability by moving the data in scan line increments instead of in whole row jumps, thus creating the effect of a sheet of paper slowly being moved through the viewing area. One system restriction of providing the smooth scrolling feature is that the rate at which new rows of data can be received is limited to the rate at which the display is being moved. Thus, successive line feeds must be separated by a minimum number of real or dummy 'fill' characters in order to allow the display to keep up with the received data. The number of fill characters will be a function of the number of scan lines per character row, the scroll rate, and the communications line speed, and may also be effected by the inclusion of software and/or hardware features such as a data buffer in the system design.

When using the CRT chip set, smooth scrolling can be implemented in software only, or with a combination of hardware and software.

Software Only Method

The software only method of smooth scrolling uses the 2672's capability to program the scan lines per row and the vertical back porch, and the ability to program an interrupt to occur at any row by programming the row value into the split screen register, IR10. Three limitations of this method are:

1. Scrolling must be in an upward direction.
2. The screen area that is scrolled must start at the top of the display but can end at any row.
3. The minimum scrolling increment is two scan lines.

The flow chart in Figure 1 shows the steps necessary for scrolling starting at the top of the screen and ending at character row 23[1], with an additional non-scrolling row at row 24. The IR10 value is set to 23, to cause an interrupt to occur at row 23. This interrupt routine then enables the line zero interrupt, so that another interrupt occurs at row 24. The line zero (row 24) interrupt routine disables the line zero interrupt so that another line zero interrupt is not asserted until the split screen (row 23) interrupt routine enables it again.

When a scroll is desired, the system software changes the screen start address to the new value (normally an address 'n' characters higher than the previous value, where 'n' is the number of characters per row) and sets a scroll flag. The top row disappears and normally the display would jump up by one row. However, the line zero interrupt routine notes that the scroll flag is set and adds the value 's-2' (s = scan lines/row) to the vertical back porch (iR3). This causes the next active screen display to begin 's-2' lines later than normal and gives the effect of the display moving up two scan lines instead of jumping a full row. If nothing else were changed, however, the bottom of the display would move down the same number of scan lines. Thus, during the split screen interrupt routine the number of scan lines per character row (IR0) is changed to two, causing only the first two scan lines of that row (which is the new

[1] Rows are numbered consecutively starting with row 0. Thus, row 23 is the twenty-fourth row of characters.

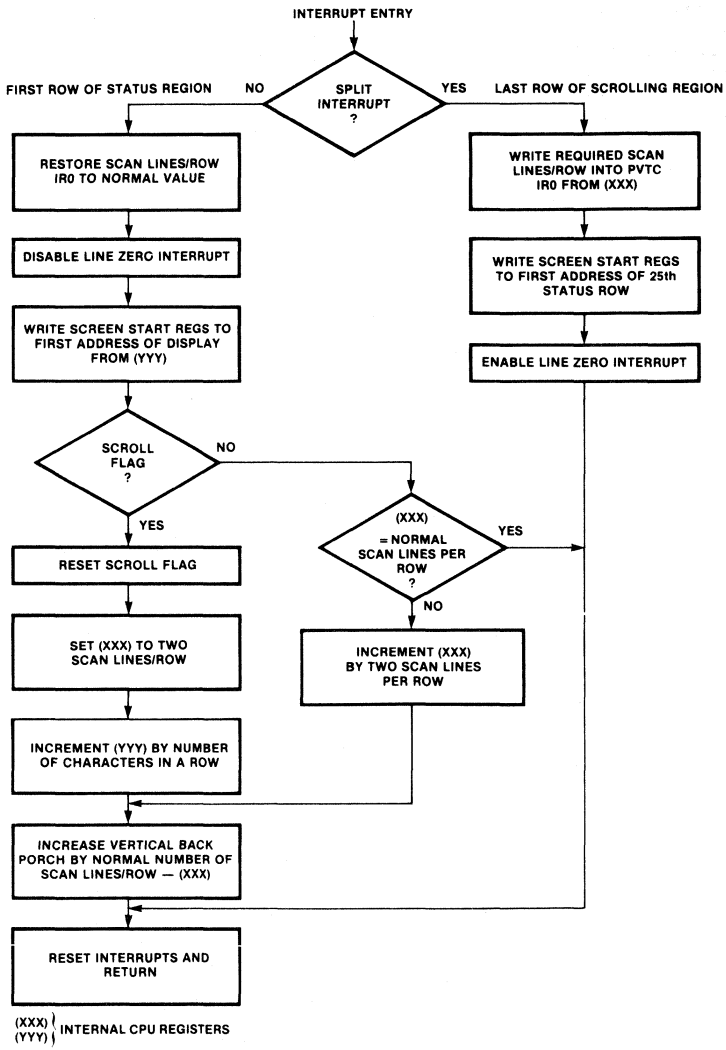


Figure 1. Smooth Scroll Flow Chart

row) to be displayed and maintaining the proper value for the total scan lines. The next line zero interrupt restores the lines per row value back to its normal state to display the whole of the last character row, and now decreases the vertical back porch increment by two. The display moves up two more scan lines and at the next split screen interrupt four scan lines are allowed. The process continues in this manner until the scroll is completed.

Note that the CPU must complete the split screen interrupt routine within two scan line times.

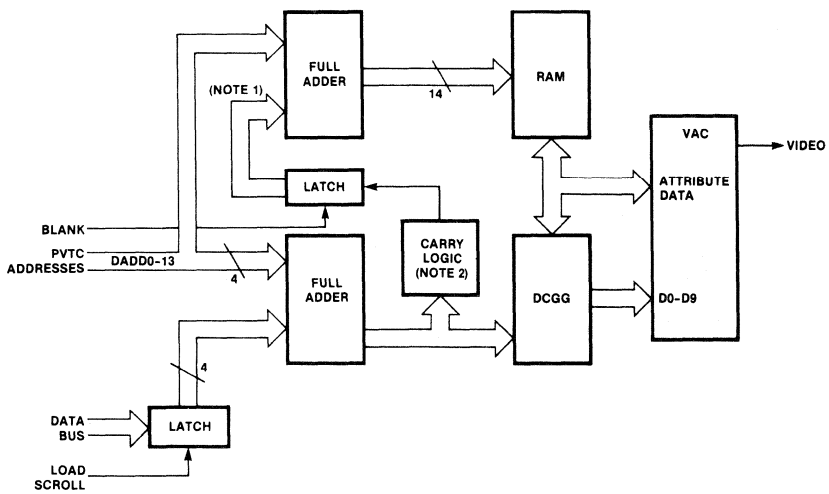
Hardware/Software Method

The hardware/software method of smooth scroll removes the restrictions of the software only method. The scrolling region can begin and end at any character row, scrolling can be performed in either the up or down direction, and the minimum scroll increment can be one scan line.

A block diagram of the required hardware for the case of full screen scroll is illustrated in Figure 2. When scrolling is required, the CPU writes the number of scan lines to scroll into the 4-bit latch during the vertical retrace interval. The scan line adder detects when the sum of the offset and the scan line address is greater than the programmed number of scan lines per row and causes the RAM address to be automatically offset to the next character row by the n-bit RAM address adder. The CPU adjusts the value in the scan lines to scroll latch at desired intervals, e.g., each vertical retrace, to effect the smooth scroll. When the scroll is completed, the screen start address is changed to the new value required for the top character row.

If only a partial screen scroll is required, the hardware must be modified to cause the offset to be applied only during the scrolling area. The lines to scroll latch of Figure 2 is replaced by the circuit shown in Figure 3. The software is written with split screen interrupts at the row before the first row which is to be scrolled (interrupt 1) and at the last scrolled row (interrupt 2). The required program actions are as follows:

1. During vertical retrace, the screen start address for the non-scrolled region at the top of the screen is loaded and IR10 is loaded with the row number required for interrupt 1.
2. During interrupt 1, a new screen start address for the scrolled region is loaded (if required), IR10 is loaded with the value required for interrupt 2, and the lines to scroll latch is loaded with five bits of data consisting of the four-bit lines to scroll value plus an asserted 'scroll' bit. The hardware will cause the scan line offset to be applied to the adder at the beginning of the next character row, as required.
3. During interrupt 2, a new screen start address for the non-scrolled region at the bottom of the screen is loaded (if required), and the 'scroll' bit is negated. The hardware will cause the scan line offset to be removed from the adder at the beginning of the next character row, as required. If the scroll region extends to the bottom of the visible screen, this interrupt is not required. Note that the time required to service this interrupt will determine the minimum number of scan lines per scrolling increment.



NOTES:
1. Hardwire adder inputs to number of characters per row.
2. Carry logic asserts output when output of adder exceeds normal scan lines/row.

Figure 2. Smooth Scroll Hardware Block Diagram

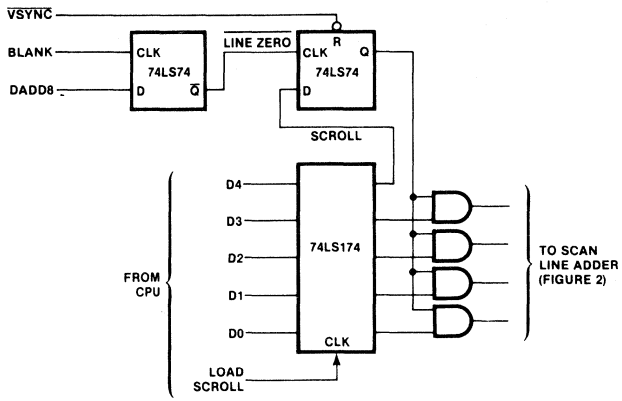


Figure 3. Partial Screen Smooth Scroll Modification

HORIZONTAL SCROLL

Horizontal scrolling allows the terminal to be used to read or write pages which are wider than the actual screen width. For example, if an actual page width of 132 characters is to be displayed on a terminal with a capacity of 80 characters per row, horizontal scrolling can be used to display any desired 80-character 'window' of the 132-character row. The window can be moved in response to operator keyboard commands, allowing all 132 columns to be observed.The CRT set capabilities allow horizontal scrolling in single or multiple character increments to be implemented using software only. As described in the 2672 data sheet, changing the contents of the screen start address register during a particular row, say row 'n', will cause the display of the next row, 'n + 1', to begin from the new address. This feature, together with the capability to interrupt at every row via the line zero interrupt, can be used to update the contents of the screen start register once each row to effect the horizontal scroll. The software operations required are as follows (see Figure 4):

1. During the vertical retrace interval, the screen start register is initialized with the starting memory address of the display page plus the desired horizontal scroll (in characters).

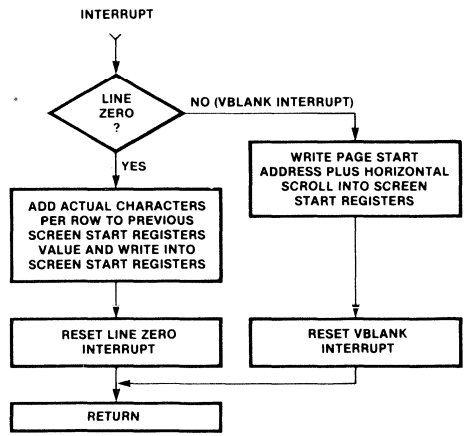


Figure 4. Horizontal Scroll

2. The line zero interrupt is enabled. During each interrupt service the value in the screen start address is incremented by the actual (not display) page width. This may be done either by referencing a table of starting addresses or by performing the required addition.

COLOR DISPLAY INTERFACE

Figure 5 illustrates the block diagram of a color monitor interface. Eight colors for foreground and background with three attributes are supplied. The system operates in the character attribute mode with a 16-bit word of data for each character: seven bits for character select, six bits for color select, and three bits for other attributes.

The two 74LS374s delay the color information by two CCLKs to allow for the two CCLK delays of dot data through the DCGG and VAC. The video output of the VAC selects foreground or background color (active dot or not) for each character cell via the 74LS157 multiplexer. A variable CCLK delay may be required to synchronize the delay of the color information through the latches to the delay of the video data from the VAC.

EXTERNAL VIDEO SYNC

Some applications require overlaying of characters on an existing video display. An example of this is the addition of subtitles to a picture display. Figure 6 illustrates a simple technique of externally synchronizing the 2672 PVTC to an external video source. The dot clock to the 2673 VAC is stopped (character clock falling edge) at the start of the PVTC's sync interval and restarted upon occurrence of the external sync signal. The sync timing programmed in the 2672 must be slightly faster than the external sync rate.

SCAN LINE COUNT GREATER THAN 16

Certain applications may require more scan lines than the 16 scan lines per character row (non-interlaced) which the 2672 can provide. Figure 7 shows the hardware required to obtain up to 32 scan lines per character row. The PVTC must be programmed for double height character rows. This causes the scan line count outputs from the 2672 (DADD4-DADD7) to increment once every two scan lines. The external flip-flop toggles each scan line to provide a fifth bit of scan line count information for the character generator. The technique permits any even value of scan lines per character row from 2 to 32 to be obtained.

BIT MAPPED GRAPHICS

Figure 8 illustrates an implementation of a bit mapped display with the chip set. In this configuration, the contents of the memory will be displayed without character generator translations. Thus, each bit in the memory corresponds to a single pixel on the display. Each horizontal scan line is defined by a contiguous set of bytes in the RAM. The data is written in groups of 8 bits by the CPU and is accessed by the PVTC in groups of 8 bits. The character generator of a normal alphanumeric configuration is replaced by an 8-bit latch to implement the one CCLK delay normally provided by the 2670. The PVTC can be programmed for one scan line/row to cause the memory addressing to proceed without repetition of addresses in each row.

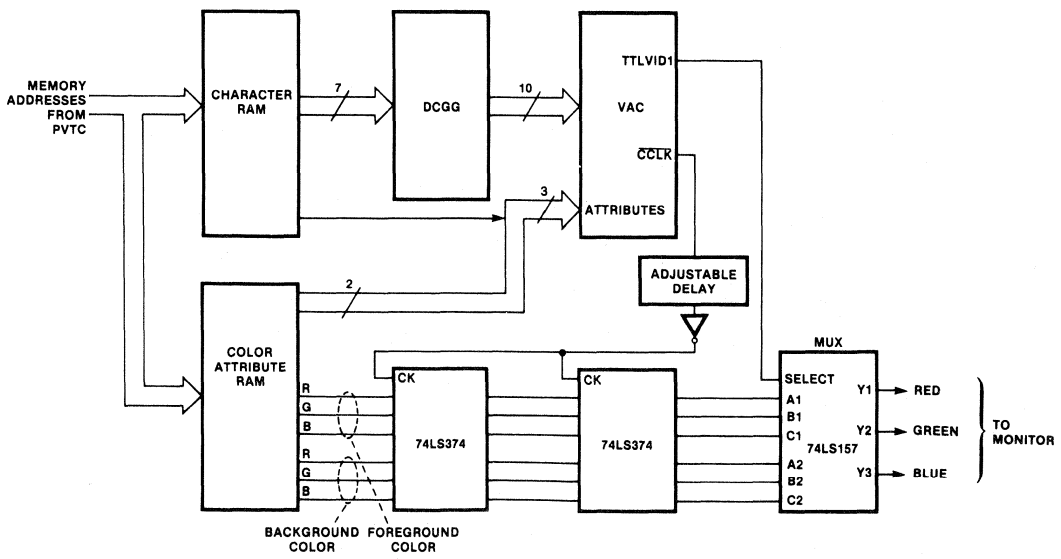


Figure 5. Color Display Block Diagram

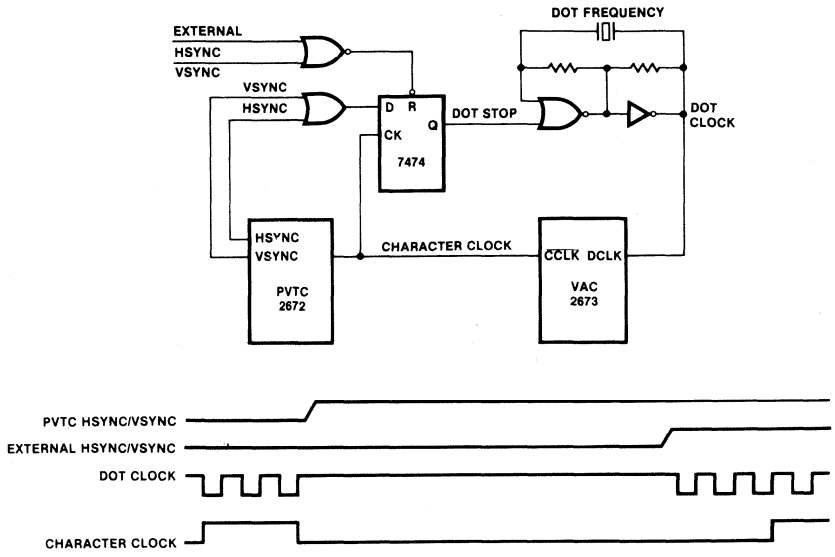


Figure 6. External Sync Lock

The PVTC can be programmed to a maximum of 256 characters/row and 128 rows/screen. Thus, a 2048 by 128 bit map is possible. If more than 128 dots are required vertically, the PVTC can be programmed for more than one scan line/row and the scan line outputs can be used as part of the RAM address, creating several segments of memory. For example, if 256 lines are required, the PVTC must be programmed for two lines/row. The use of LA0 as part of the memory address creates two segments. The CPU writes the data for odd scan lines in one segment of the memory and the data for even scan lines in the other. As the PVTC accesses the RAM the scan line count will go from 0 to 1 addressing the even and then odd portion of the RAM for each character row.

Figure 8 shows the DCGG as optional. By using the 2672's split screen capability and changing 2672 parameters, the CPU can enable the DCGG to be active for a portion of the screen thereby incorporating both bit-mapped and alphanumeric sections on the same display.

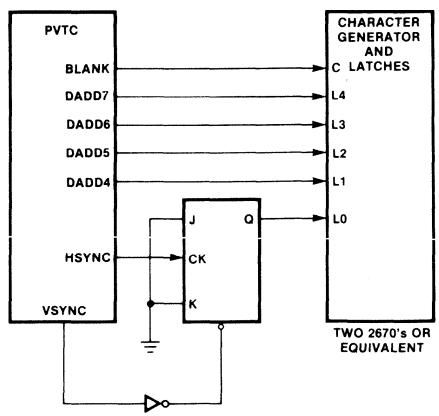


Figure 7. Greater Than 16 Scan Lines Per Row

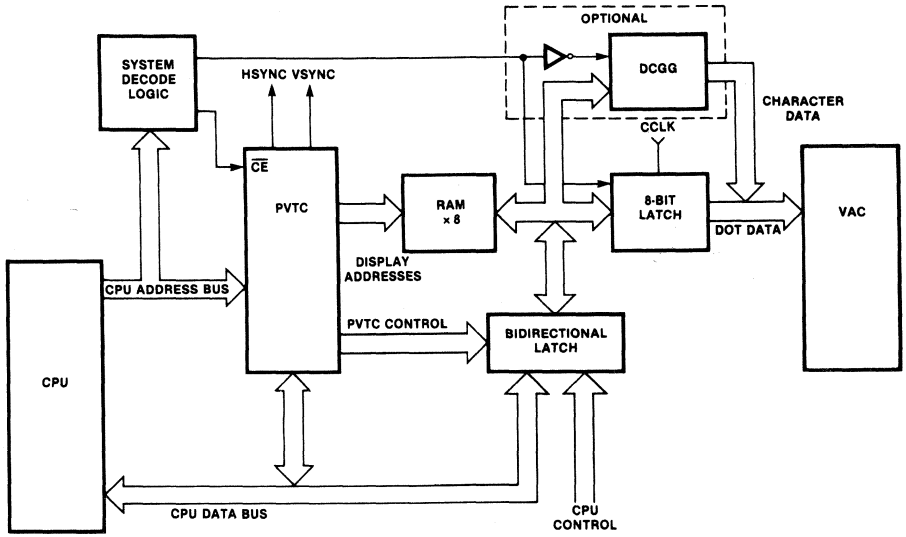


Figure 8. Implementation of Bit Mapped Display

SMOOTH SCROLLING WITH THE SCN2674 AVDC

Originally published by Signetics May 1983

SMOOTH (SOFT) SCROLLING

Scrolling is used in terminals to provide the effect of an endless page on which data can be written. In traditional implementations, once the screen fills up, space for the next row of data is made available by removing the top row and moving all the remaining rows up by one row, thus creating a blank row at the bottom of the screen where the new information can be placed. (The process can be reversed if the new data is to be written at the top of the screen.) Although this is perfectly satisfactory when the operator is using the terminal to enter data, the technique results in a readability problem when viewing data which is being received from an external source at a relatively high speed, since the rows are jumping up (or down) at a fast rate. Smooth or soft scrolling improves readability by moving the data in scan line increments instead of whole row jumps, thus creating the effect of a sheet of paper slowly being moved through the viewing area.

NOTE: One system restriction in providing smooth scrolling is that the rate at which data can be received is limited to the rate at which the display is being moved. Thus, successive line feeds must be separated by a minimum number of real or 'fill' characters in order to allow the display to keep up with the received data. The number of fill characters is a function of the scan lines per row, the scroll rate, and the communications line speed, and may also be affected by software and/or hardware features such as a data buffer in the system design. Some terminals use a buffer to store received data while scrolling takes place and an 'X-ON/X-OFF' protocol to advise the sender to start and stop sending as the buffer fills and empties.

The SCN2674 Advanced Video Display Controller (AVDC) provides dedicated on-chip hardware to offload the smooth scrolling task from the CPU, but allows the user to choose several variables about the scrolling region. Two AVDC registers (IR12[6:0] and IR13[6:0]) are programmed to select the size and position of the smooth scrolling region, while a third (IR14[3:0]) specifies the number of scan lines to be scrolled at that time. The split screen facilities of the AVDC can be used in conjunction with the smooth scroll logic to cause a split to a new address at the beginning or end of the scrolling region.

Since the scrolling region is normally fixed, the registers which specify it are usually programmed only during the chip initialization routine. To effect a smooth scroll, the CPU merely sets two control bits (IR12[7] and IR13[7]) in the AVDC initialization register group and specifies the number of lines to scroll in the lines to scroll register (IR14[3:0])[1]. By changing the latter value periodically, the CPU can control the scrolling speed and direction. For example, setting lines to scroll to one initially and incrementing it by one during every vertical retrace interval will cause the display to scroll up at 60 scan lines per second (assuming the AVDC is programmed for 60Hz vertical rate), while incrementing the value during every other retrace will reduce the scrolling speed to 30 lines per second. Similarly, if lines to scroll is set to two initially and is incremented by two during every vertical retrace interval, the display will scroll up at 120 scan lines per second. Since the speed is controlled by the CPU, it can even automatically adjust it as a function of the number of characters received in the last row, speeding up the rate for short lines and slowing it down for long lines.

While smooth scrolling is taking place, the number of character rows is increased by one for each scrolling region, although the total number of scan lines remains the same. The AVDC automatically inserts the row as required, so that the user need not change the value programmed in the character rows per screen register. In order to minimize changes to program parameters while smooth scrolling is in effect, the partial row added at the bottom of the display does not have a row 'number'. For example, if the bottom of the scrolling region is programmed at row 20 in IR13[6:0], the numbers of the rows following the partial row at the bottom of the display will begin with 21. Thus, if an interrupt is programmed to occur at row 25, it occurs at the same logical row whether scrolling is in effect or not. Since in many cases an interrupt or an automatic split is required at the bottom of the scrolling region, the AVDC's internal logic automatically changes the operation of the split screen 2 interrupt and the automatic split enabled by the SPL2 bit of screen start register 2 (SSR2) while smooth scrolling is in effect.

Considering the example above, if split screen 2 interrupt is enabled, it will occur at the beginning of the first scan line of row 20 when scrolling is not taking place, but will be delayed until the first scan line of the partial row following row 20 when scrolling is taking place. If an automatic split to the contents of SSR2 is programmed by asserting the SPL2 bit, the split will occur at the beginning of row 21 whether or not scrolling is taking place (row 21 being the row following the partial row when scrolling is on).

NEW ROW BLANKING

The fact that an extra row is displayed during smooth scrolling increases the amount of memory required to hold the screen image. Thus, a 25 row by 80 character display which normally requires 2,000 bytes of video memory will require 2,080 bytes while smooth scrolling is in effect. If sufficient memory is not available, a portion of the video RAM will be re-scanned, causing characters to be displayed twice on the screen. As an example, assume the following parameters have been programmed for a 25 x 80 display:

IR9[7:4] = H'1' (buffer last address = 2,047)
IR8[7:0] = H'30' (buffer first address = 48)
IR12[6:0] = H'00' (first scrolling row)
IR13[6:0] = H'18' (last scrolling row)

This makes only 2,000 bytes available as the video RAM. In this case, when scrolling is in progress, the wraparound feature of the AVDC will cause the bottoms of the characters to be displayed on the top row while the tops of the same characters are displayed on the bottom row.

The AVDC provides a 'new row blanking' feature which may be optionally used if the extra memory which is required is not available. If LZ UP (IR11[6]) is asserted, the scan line number outputs of the AVDC will be forced to zero for each scan line in the partial row at the bottom of the screen. If the terminal's character generator provides blanks during scan line zero, the partial row will be automatically blanked on the display. LZ DOWN (IR11[7]) can be used to produce the same effect at the top partial row. When the scroll is completed, the new data to be shown can be loaded into the video RAM and the bits are then negated. However, the effect of this will be that a blank row smoothly scrolls onto the screen and is then rapidly updated with new data, instead of the new data smoothly scrolling onto the display. The best solution, of course, is to provide the extra memory which is required.

[1]Note that the number of lines to scroll is independent of whether the character row being scrolled is single height or double height. Thus, double height characters can be scrolled as easily as single height characters and scroll at the same rate.

PROGRAMMING EXAMPLES

Figures 1 through 4 illustrate some of the smooth scrolling effects which may be implemented with the 2674 AVDC, and figures 5 through 8 are flow charts detailing the basic software functions required to achieve them. The following terms are used in the flow charts:

Acronym	Definition
TROW	Row number of the top row of the scrolling region
BROW	Row number of the bottom row of the scrolling region
SSADR	Address of the first character on the screen (row 0, column 0)
TOPADR	Address of the first character of the scrolling region (row TROW, column 0)
SPLADR	Address of the first character of the first row after the scrolling region (row BROW + 1, column 0)
N	Scroll speed constant (slow, N = 1; medium, N = 2; fast, N = 3)

For the examples dealing with two scrolling regions, the suffixes (U) and (L) are appended to the acronyms above to indicate upper and lower regions respectively.

Figures 1 and 2 illustrate displays with a single scrolling area. In figure 1, the scrolling area starts at the top of the screen and ends near the bottom, with a status row or rows following. Figure 2 shows a similar case, but with status areas above and below the scrolling area.

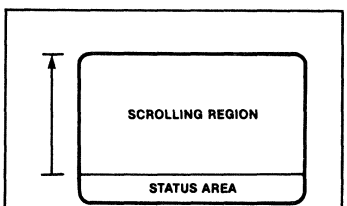


Figure 1. Single Scrolling Region With One Status Area

Even more sophisticated scrolling schemes can be implemented by combining the smooth scrolling capabilities of the AVDC with interrupts. Consider, for example, a system which requires two scrolling regions. The CPU sets up the parameters for the topmost scrolling region

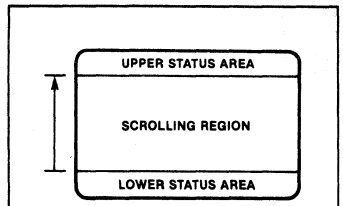


Figure 2. Single Scrolling Region With Two Status Areas

(start scroll row, end scroll row, number of lines to scroll) during the vertical retrace interrupt. The AVDC is also programmed to issue an interrupt at the last row of the scrolling area. The service routine for this interrupt then reprograms the appropriate registers with the values required for the second scrolling region.

In the case of figure 3, the two scrolling regions are scrolling simultaneously in the same direction and at the same rate. Therefore it is only necessary to change the TROW and BROW values and the corresponding split addresses TOPADR and SPLADR to those required for the bottom

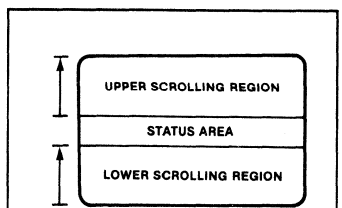


Figure 3. Two Scrolling Regions With Simultaneous Scrolling

scrolling region. It is even possible to simultaneously scroll the two regions at different rates, or to scroll the two regions in different directions, as illustrated in figure 4. Essentially, these two regions become totally independent scrolling regions. For this case, the interrupt which occurs during the last row or partial row of the topmost scrolling region is used to indicate the end of the topmost scrolling

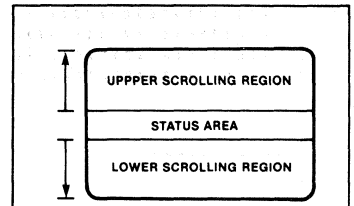


Figure 4. Two Scrolling Regions With Independent Scrolling

region. The AVDC hardware would allow the TROW and BROW parameters to be changed at this time but not the other scrolling parameters. Thus, the interrupt routine here only asserts the line zero interrupt enable bit, causing an interrupt at the first scan line of the first row following the scrolling region. When that interrupt is serviced, the registers are free to be changed and the appropriate scrolling parameters for the ensuing scrolling region (some of which were precalculated during the vertical retrace interrupt) may now be set up.

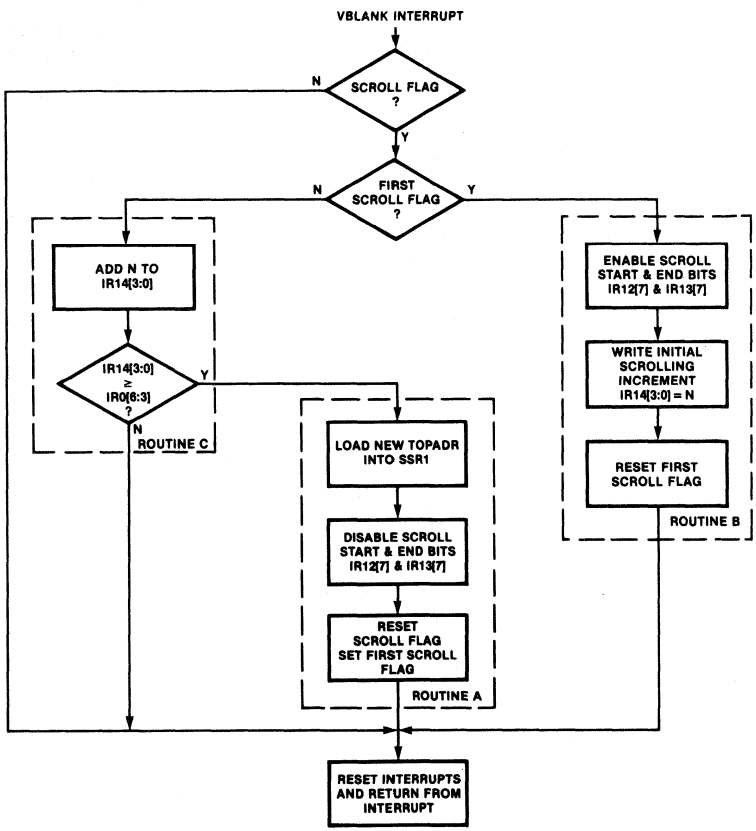
SOFT SCROLLING IN ROW TABLE ADDRESSING MODE

When the AVDC operates in row table addressing mode, each character row on the screen has its own starting address in a row table, a table of addresses residing in either the CPU's memory or in the display memory. Smooth scrolling may also be used in this mode, but some of the operations which are done automatically by the AVDC when it is in the sequential addressing mode must be accounted for by modifications to the row table while scrolling is in effect. In essence, the split screen capabilities utilized in the prior examples must be incorporated into the row table function.

As described previously, an extra (partial) row is displayed while smooth scrolling takes place. Therefore, the table of starting addresses must be updated to include the starting address of the partial row being scrolled onto the screen. This is similar to the procedure required for a character row insertion. When scrolling up, the new table entry is inserted after the entry for what was the last row of the scrolling area, and for scrolling down, the

new table entry is inserted prior to the entry for what was the first row of the scrolling area. When scrolling is completed, the entry for the row which was scrolled off the display must be removed from the list and the remainder of the list must be pushed up to fill the empty place in the table.

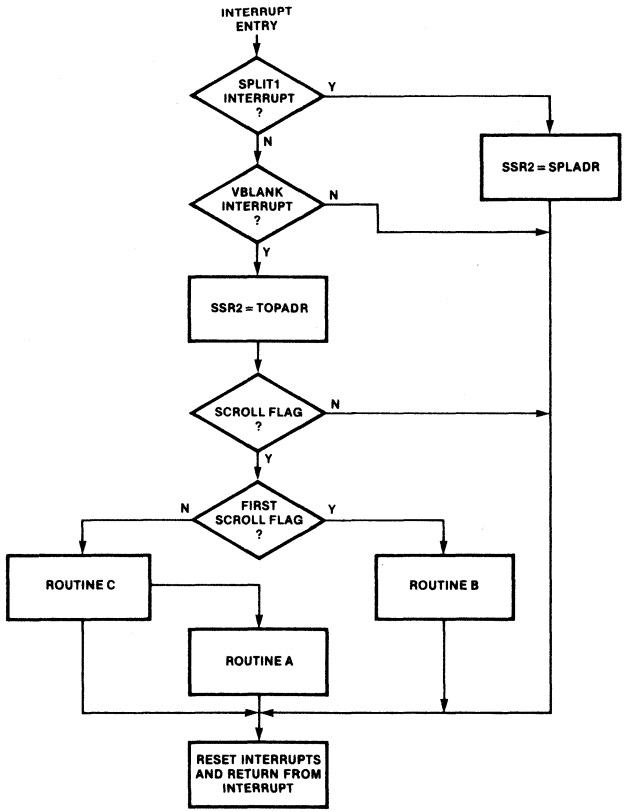
The scrolling flow charts in figures 5 through 8 may be adapted for row table addressing mode. To illustrate the required changes, figure 5 has been modified and is shown in figure 9.



NOTES:

1. AVDC initialization routine loads IR12[6:0] with '00', IR13[6:0] with BROW, SSR2 with SPLADR, asserts the SPL2 bit of SSR2 and the FIRST SCROLL flag, and enables the VBLANK interrupt.
2. SSR1 is initially loaded with SSADR, which in this case is the same as TOPADR.
3. Mainline program asserts the SCROLL flag to begin the scrolling sequence.

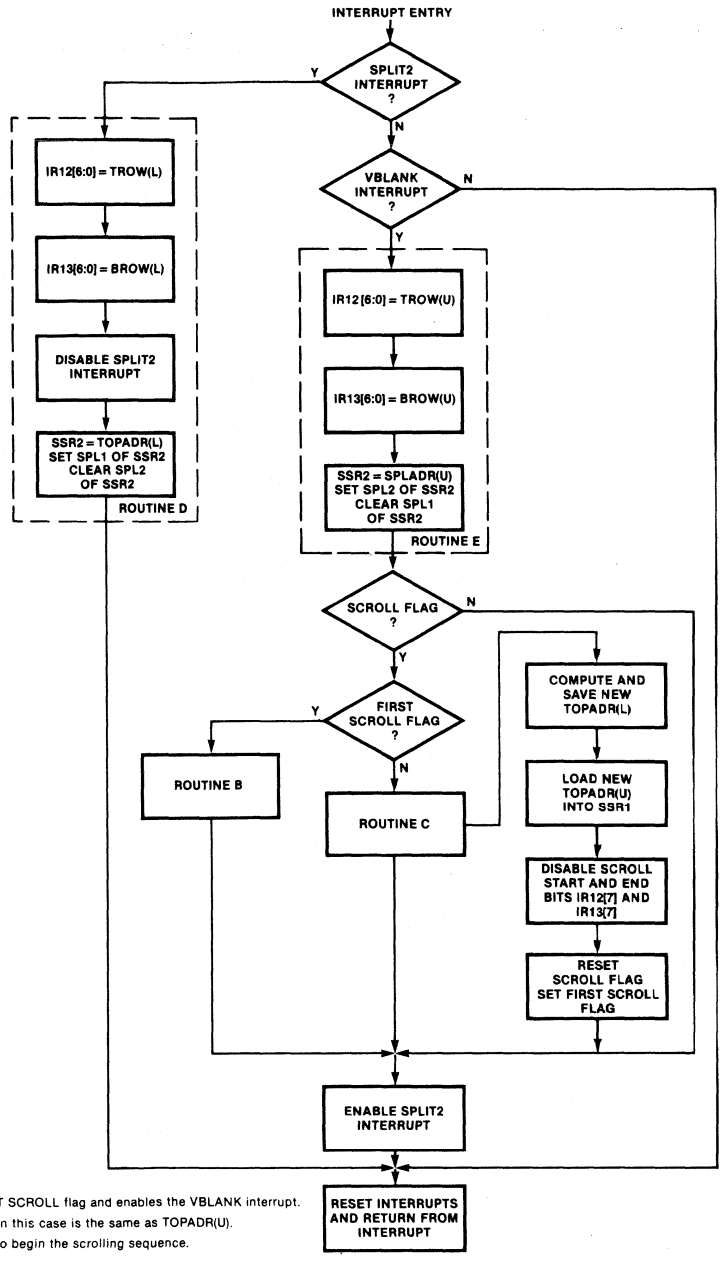
Figure 5. Flow Chart For Figure 1



NOTES:

1. AVDC initialization routine loads IR12[6:0] with TROW, IR13[6:0] with BROW, SSR1 with SSADR, asserts the FIRST SCROLL flag, and enables the SPLIT1 and VBLANK interrupts.
2. The SPL1 and SPL2 bits of SSR2 must be asserted each time SSR2 is loaded.
3. Mainline program asserts the SCROLL flag to begin the scrolling sequence.

Figure 6. Flow Chart For Figure 2



NOTES:

1. AVDC initialization routine asserts the FIRST SCROLL flag and enables the VBLANK interrupt.
2. SSR1 is initially loaded with SSADR, which in this case is the same as TOPADR(U).
3. Mainline program asserts the SCROLL flag to begin the scrolling sequence.

Figure 7. Flow Chart For Figure 3

NOTES:

1. AVDC initialization routine asserts the FIRST SCROLL flag and enables the VBLANK interrupt.
2. SSR1 is initially loaded with SSADR, which in this case is the same as TOPADR(U).
3. Mainline program asserts either the TOP SCROLL flag or the BOTTOM SCROLL flag, or both, to begin the corresponding scrolling sequence.

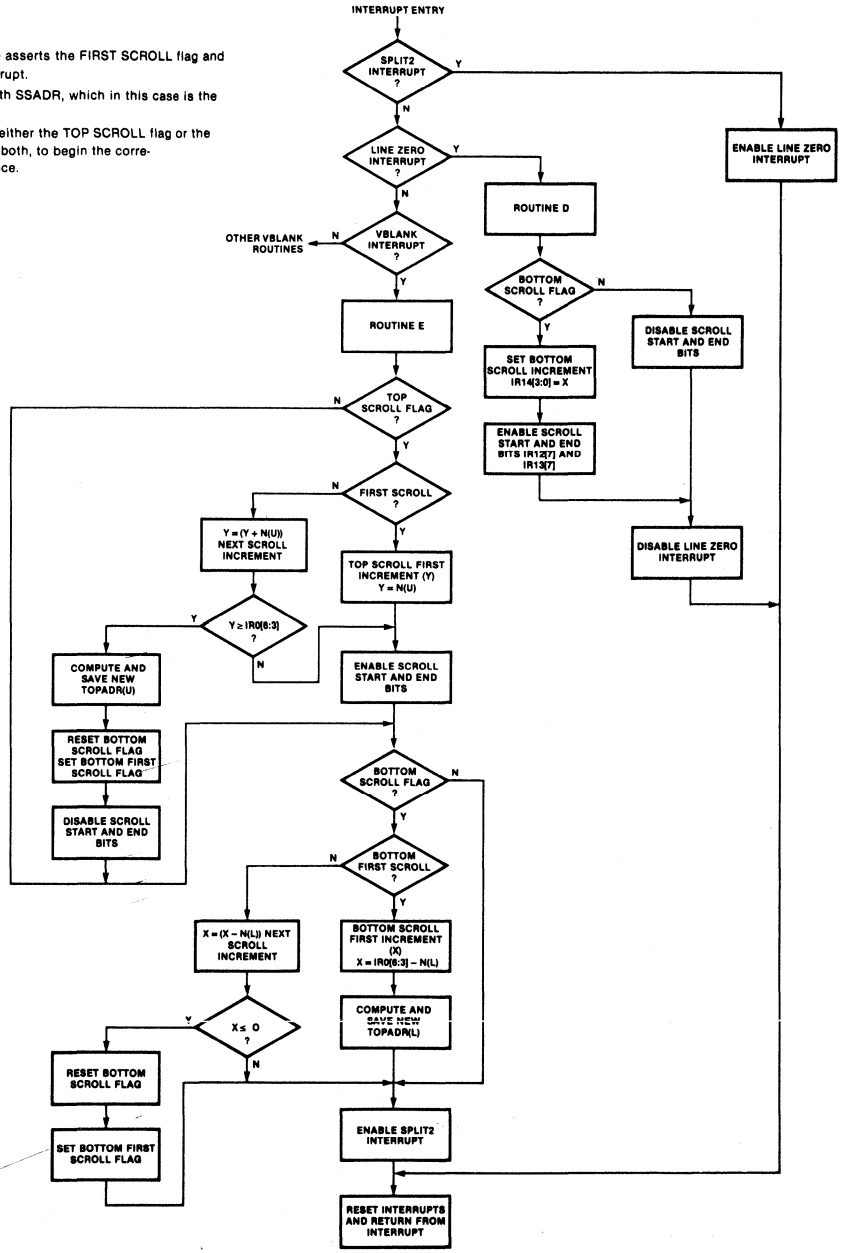
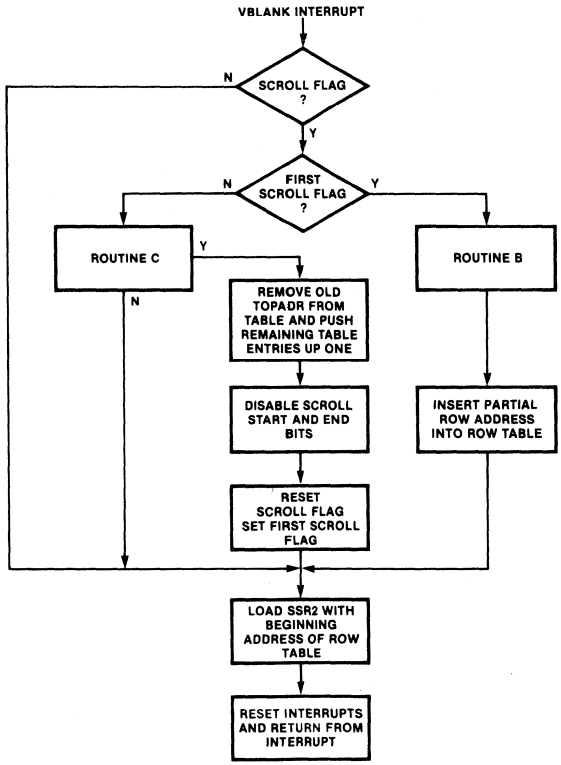


Figure 8. Flow Chart For Figure 4



NOTES:

1. AVDC initialization routine loads IR12[6:0] with '00', IR13[6:0] with BROW, SSR2 with the starting address of the row table, asserts the FIRST SCROLL flag, and enables the VBLANK interrupt.
2. SSR1 is initially loaded with SSADR, which in this case is the same as TOPADR.
3. Mainline program asserts the SCROLL flag to begin the scrolling sequence.

Figure 9. Flow Chart For Figure 1 With Row Table Addressing

VIDEO GAMES

VOICE SYNTHESIZER

GENERAL DESCRIPTION

The MEA8000 is a 24-pin N-MOS integrated circuit for generating good quality speech from digital code with a programmable bit rate. The circuit is primarily intended for applications in microprocessor controlled systems, where the speech code is stored separately.

Features

- Interfaces easily with most popular microprocessors and microcomputer
- 8-bit wide data bus
- 32-bit wide data buffer holding speech frame codes
- Digital filter of 8th order with 3 programmable formant frequencies, one fixed formant frequency, and 4 programmable formant bandwidths
- Programmable amplitudes
- Programmable duration of each frame; 8, 16, 32 or 64 ms
- Synthesis occupies less than 1% of control processor time
- Capable of sophisticated unvoiced sound generation
- Crystal controlled oscillator or external (TTL) clock
- Minimal external audio filter requirement
- Single + 5 V power supply

QUICK REFERENCE DATA

parameter	condition	symbol	min.	typ.	max.	unit
Supply voltage	pin 13	V_{DD}	4,5	5,0	5,5	V
Supply current	no audio load	I_{DD}	–	30	50	mA
Inputs						
Input voltage	HIGH	V_{IH}	2,0	–	V_{DD}	V
Input voltage	LOW	V_{IL}	−0,5	–	0,8	V
Input capacitance		C_I	–	–	7	pF
Outputs						
Output voltage	$-I_{OH}$ = 100 μA	V_{OH}	2,4	–	–	V
Output voltage	I_{OL} = 1,6 mA	V_{OL}	–	–	0,4	V
Capacitance		C_L	–	–	30	pF
Operating ambient temperature range		T_{amb}	0	–	+ 70	°C

PACKAGE OUTLINE

24-lead DIL; plastic (SOT-101A).

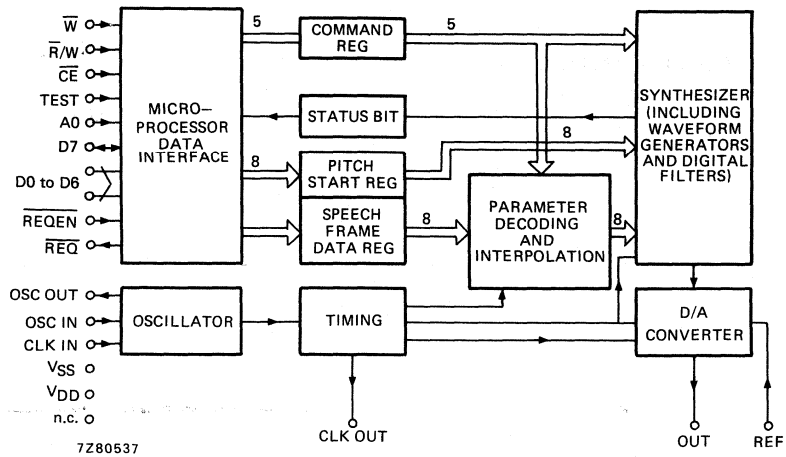


Fig. 1 Block diagram.

PINNING

1	V_{SS}	ground
2	$\overline{REQ}$	data request
3	D7	data bus
4	D6	
5	D5	
6	D4	
7	D3	
8	D2	
9	D1	
10	D0	
11	A0	data/control input
12	$\overline{CE}$	chip enable
13	V_{DD}	supply voltage
14	$\overline{REQEN}$	request enable input
15	N.C.	not connected
16	OSC IN	internal oscillator
17	OSC OUT	internal oscillator
18	CLK IN	clock input
19	REF	reference current
20	OUT	speech output
21	CLK OUT	internal clock output
22	$\overline{R}$/W	read/write
23	$\overline{W}$	write
24	TEST	test use only

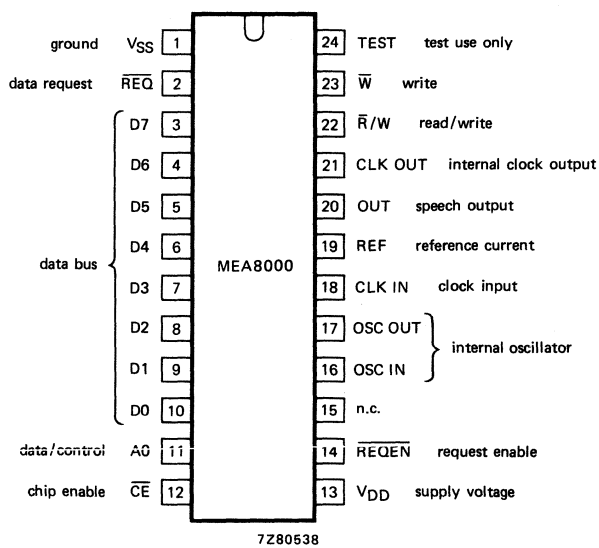


Fig. 2 Pinning diagram.

FUNCTIONAL DESCRIPTION (pin number)

Control

D0 to D7	(10 to 3)	Data bus to which command or speech can be written.
D7	(3)	Data port via which the status can be read.
$\overline{CE}$	(12)	Chip enable (chip select).
$\overline{W}$	(23)	Write.
$\overline{R}$/W	(22)	Read/Write The control signals $\overline{W}$ and $\overline{R}$/W allow connections to most microcomputers or microprocessors (see timing diagrams).
A0	(11)	Data/control input: discriminates between speech code input buffer (A0 = '0') and command register (A0 = '1') during a 'write' operation.
$\overline{REQ}$	(2)	Data request (open drain output); output signal which follows inverse of the status REQ bit, but only if enabled by either the ROE bit in the command register or the external $\overline{REQEN}$ pin.
$\overline{REQEN}$	(14)	Request enable input; $\overline{REQEN}$ = '0' enables the status REQ output, independent of the status of the command register.

Timing

OSC IN	(16)	Connections for internal clock oscillator; nominal crystal frequency 4 MHz.
OSC OUT	(17)	
CLK IN	(18)	Clock input for external clock, TTL compatible, 4 MHz.
CLK OUT	(21)	A buffered output for the internal clock cycle (which is equal to CLK divided by 3). May be used as a clock, for a microprocessor, for example.

Output

REF	(19)	Input pin for biasing the audio output level. This reference current can be derived from a resistor to the positive supply.
OUT	(20)	Speech output; this output is a 64 kHz pulse, modulated in both width and amplitude. It is configured as a current sink with a saturating voltage of about 3 V.

Supply

V_{DD}	(13)	Single supply voltage, nominally 5 V, but battery operation is possible.
V_{SS}	(1)	Ground.
TEST	(24)	Used for testing purposes. Changes other pin functions. Must be tied to ground for user operation.
NC	(15)	It is recommended to ground this pin.

HANDLING

Inputs and outputs are protected against electrostatic charge in normal handling. However, to be totally safe, it is desirable to take normal precautions appropriate to handling MOS devices (see 'Handling MOS Devices').

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

parameter	conditions	symbol	min.	max.	unit
Supply voltage range		V_{DD}	−0,5	+ 7	V
Voltage with respect to V_{SS}	on any pin	V_I	−0,5	+ 7	V
Output voltage	pins 2 and 20	V_{REQ}, V_{OUT}		15	V
Storage temperature range		T_{stg}	−20	+ 125	°C
Operating ambient temperature range		T_{amb}	0	+ 70	°C

CHARACTERISTICS

T_{amb} = 25 °C; V_{DD} = 5 V, unless otherwise specified; all voltages referenced to V_{SS}

parameter	conditions	symbol	min.	typ.	max.	unit
Symbol						
Supply voltage	note 1	V_{DD}	4,5	5,0	5,5	V
Supply current	no audio load	I_{DD}	–	30	50	mA
Inputs						
D0 to D7, A0, $\overline{CE}$, $\overline{W}$, $\overline{R}$/W, $\overline{REQEN}$, CLK IN						
Input voltage HIGH		V_{IH}	2,0	–	V_{DD}	V
Input voltage LOW		V_{IL}	–0,5	–	0,8	V
Input leakage current	note 2	I_{IR}	–	–	10	μA
Input capacitance		C_I	–	–	7	pF
Outputs						
D7 (I/O), CLK OUT						
Output voltage HIGH	$-I_{OH}$ = 100 μA	V_{OH}	2,4	–	–	V
Output voltage LOW	I_{OL} = 1,6 mA	V_{OL}	–	–	0,4	V
Output load capacitance		C_L	–	–	50	pF
$\overline{REQ}$						
Output voltage HIGH	open drain	V_{OH}	–	–	13,2	V
Output voltage LOW	I_{OL} = 1,6 mA	V_{OL}	–	–	0,4	V
Output load capacitance		C_L	–	–	50	pF
Audio output						
Reference current	pin 19; note 8	I_{REF}	–	–	0,3	mA
Output current	pin 20; peak value					
	I_{REF} = 0 mA	I_{OUT}	–	100	–	μA
	I_{REF} = 0,1 mA	I_{OUT}	–	1,7	–	mA
	I_{REF} = 0,3 mA	I_{OUT}	–	5	–	mA
Output voltage	pin 20; for linear operation; note 3;					
	I_{REF} = 0,1 mA	V_{OUT}	2,5	–	13,2	V
Oscillator						
Crystal frequency	internal	f_{XTAL}	–		4,00	MHz
Clock frequency	external	f_{CLK}	–		4,00	MHz

TIMING CHARACTERISTICS (note 4) (Figs 6 and 7)

parameter	condition	symbol	min.	typ.	max.	unit
Write enable		t_{WR}	200	–	–	ns
Address set-up		t_{AS}	30	–	–	ns
Address hold		t_{AH}	30	–	–	ns
Data set-up for write		t_{DS}	150	–	–	ns
Data hold for write		t_{DH}	30	–	–	ns
Request hold	note 5	t_{RH}	–	–	350	ns
Request next	note 6 clock frequency = 3,84 MHz	t_{RN}	–	–	3	μs
Read enable		t_{RD}	200	–	–	ns
Data delay for read	note 7	t_{DD}	–	–	150	ns
Data floating for read	note 7	t_{DF}	–	–	150	ns
Request valid before write		t_{RV}	0	–	–	ns
Request output enable response		t_{ROE}	–	–	750	ns
→ Control set-up		t_{CS}	20	–	–	ns
→ Control hold		t_{CH}	20	–	–	ns

Notes

1. The circuit will continue to operate from a supply of up to 6,5 V, but without necessarily meeting the specification.
2. This is also valid for V_{DD} = 0 V.
3. This permits the connection of the output load to a supply higher than that supplying the synthesizer.
4. Timing reference level is 1,5 V.
5. An external pull-up resistor is required, as this is an open drain output.
   The time (t_{RH}) to reach 2,0 V is specified at a load to 5 V of 3,3 kΩ and 50 pF.
6. Between two data write operations of one speech frame.
7. Levels greater than 2,0 V for a '1' or less than 0,8 V for a '0' are reached with a load of one TTL input and 50 pF.
8. Typical voltage level at the REF pin is 2,5 V.

OPERATION PRINCIPLE

The MEA8000 has been designed for a vocal tract modelling technique of voice synthesis. This method gives the lowest possible bit rate for speech quality which is acceptable for most industrial applications.

Figure 3 shows a simplified electronic model of the human vocal tract as a formant synthesizer. A combination of a periodic signal, representing the pitch of the original speech, and an aperiodic signal, representing the unvoiced sound in the speech. Both these signals are fed to a variable filter comprising four resonantors (via an amplifier which controls the amplitude of the synthesized sound). The resonators model the sound in accordance with the formats in the original speech. Each resonator is controlled by two parameters, one for the resonant frequency and one for the bandwidth.
The information required to control the synthesizer is:

- pitch
- amplitude
- voice/unvoiced source selector

} excitation source (vocal cords)

- filter control — spectrum shaping (vocal tract)

A good replica of the original speech is obtained by periodic updating of this control information.

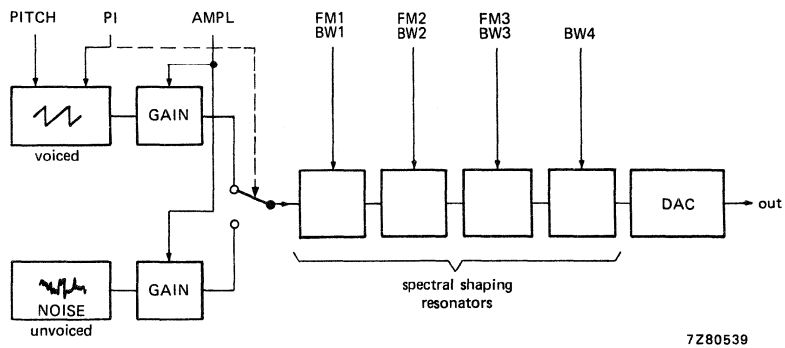


Fig. 3 Electronic model of human vocal tract.

OPERATION

Speech is generated by suitable filtering of a relatively low frequency sawtooth waveform for voiced sounds, or of random noise for unvoiced sounds. New parameters for both the digital waveform generator and the digital filter are supplied to the synthesizer in coded groups of 4 bytes via the data bus. The code group also contains the duration of the next speech frame to be produced (8, 16, 32 or 64 ms).

The output sample rate is 64 kHz or 8 times the internal sample rate with linear interpolation in between. This greatly reduces the need for an external analogue output filter.

Modes of operation

1. STOP mode: characterised by a silent output and the status $\overline{\text{REQ}}$ bit set to '1'. This mode is entered from power up or by STOP command. The mode is entered automatically if at the end of an active speech frame the next four parameter bytes are not yet received while the CONT bit in the command register is a '0'. In the latter case the final speech frame will be repeated once but with a decaying amplitude and the same pitch.
2. ACTIVE mode: a speech sample is being produced.
3. CONTINUOUS mode: entered if an active speech frame is finished and new data is not supplied in time while the CONT bit in the command register is a '1'. The synthesizer will repeat the last speech frame indefinitely until all four new data bytes are received, or a STOP command, or a reset of the CONT bit.

Speech code input buffer

Speech code is written to the synthesizer when $\overline{CE}$ and $\overline{W}$ are both '0', while $\overline{R}/W$ = '1' and A0 = '0'. Also the status $\overline{REQ}$ bit must read a '1', otherwise the synthesizer is still busy and will not react to a data write operation.
Starting from the STOP mode, the first data will be interpreted as a starting value for the PITCH. Thereafter every four successive data bytes are treated as a group of speech code. The coded speech frame format is shown in Fig. 4.

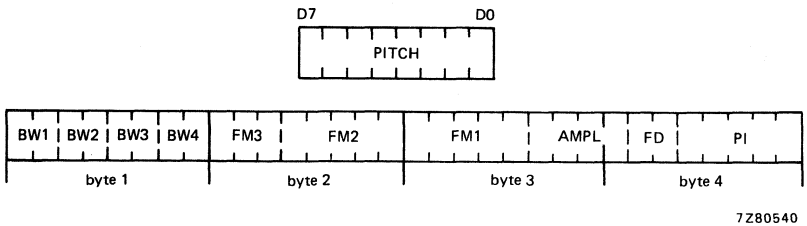


Fig. 4 Format of coded speech frame.

code	bits	parameter
PITCH	8	initial value for pitch
FD	2	speech frame duration
PI	5	pitch increment (rate of change) or noise selection
AMPL	4	amplitude
FM1	5	frequency of 1st formant
FM2	5	frequency of 2nd formant
FM3	3	frequency of 3rd formant
FM4	0	frequency of 4th formant (fixed)
BW1	2	bandwidth of 1st formant
BW2	2	bandwidth of 2nd formant
BW3	2	bandwidth of 3rd formant
BW4	2	bandwidth of 4th formant

During each data write operation, the status REQ bit will be cleared to '0'.
It appears within a few microseconds, requesting the next byte of the group.
The request for the first byte of the next group always appears shortly after the beginning of the current speech frame, and all four bytes must be provided before it finishes. This leaves the control circuit (i.e. microprocessor) enough time to use polling, instead of interrupts, as the minimum time of a speech is 8 ms.

When in the STOP mode the synthesizer will commence producing sound after receipt of 1 + 4 bytes.

Status bit

The status bit is accessed at $\overline{CE}$ = $\overline{R}/W$ = '0'.
The status of $\overline{W}$ and A0 are arbitrary.
Pin D7 reveals the request for a (next) speech code byte: '0' = busy, '1' = request for data.

Command register

A command is written to the synthesizer at $\overline{CE}$ = $\overline{W}$ = '0' while A0 = $\overline{R}$/W = '1'.

D7	D6	D5	D4	D3	D2	D1	D0
			STOP	CONT enable	CONT	ROE enable	ROE
NOT USED			'0' = INVALID '1' = STOP	00 = INVALID 01 = INVALID 10 = SLOW STOP 11 = CONTINUE		00 = INVALID 01 = INVALID 10 = DISABLE $\overline{REQ}$ OUTPUT 11 = ENABLE $\overline{REQ}$ OUTPUT	

STOP — Stop mode. This results in an immediate reset of the synthesizer to the STOP mode. The ROE and CONT are not affected by this command.

CONT — Continuous mode. This bit can be set or cleared only if the corresponding CONT enable bit is programmed as a '1'. In the continuous mode the synthesizer will not revert to the STOP mode if all four parameters are not received before the end of the current speech frame, but repeat it indefinitely.

If CONT = '1' the last frame will be repeated once with decaying amplitude and the same pitch before the stop mode is entered.

ROE — Request Output Enable. This can be set or cleared only if the corresponding ROE enable bit is a '1'. ROE determines whether the request in the status bit appears on the $\overline{REQ}$ pin.

Note: the same can be achieved by connecting the $\overline{REQEN}$ pin (request enable) to a '0'.

After power on, the command register bits CONT and ROE will both be zero. Thus power on equals the command 00011010 = 1 A (hexadecimal).

Control signals

With the three control signals $\overline{CE}$, $\overline{W}$ and $\overline{R}$/W the synthesizer is made compatible with most microprocessors and microcomputers.

$\overline{CR}$	$\overline{W}$	$\overline{R}$/W	A0		Operation
0	0	1	0		WRITE DATA
0	0	1	1		WRITE COMMAND
0	X	0	X		READ STATUS
0	1	1	X		3-STATE DATA BUS
1	X	X	X		3-STATE DATA BUS

Power supply

During (slow) power up or power down the circuit will not produce any spurious sound. As soon as the supply is high enough for reliable operation, the circuit will be in the STOP mode with ROE = CONT = '0'.

Timing diagrams

The control signals $\overline{CE}$, $\overline{R}/W$ and $\overline{W}$ have been specified to enable easy interface to most microprocessors and microcomputers. For instance, with connection to an MAB8048 microcomputer the $\overline{R}/W$ and $\overline{W}$ inputs can be used as the $\overline{RD}$ and $\overline{WR}$ strobe inputs.

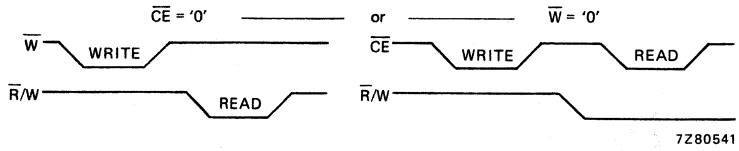


Fig. 5 Typical waveforms of the control signals.

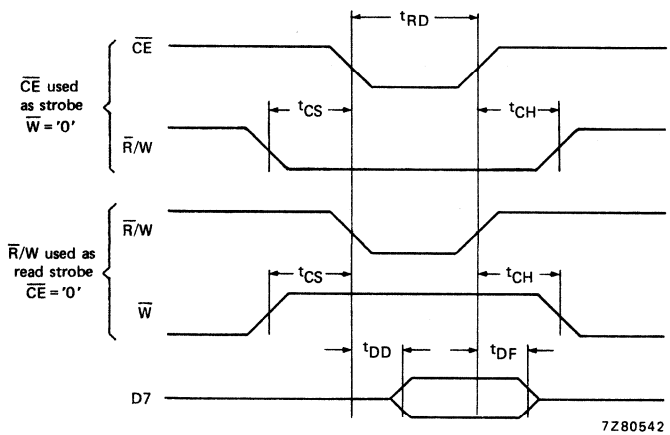


Fig. 6 Read timing.

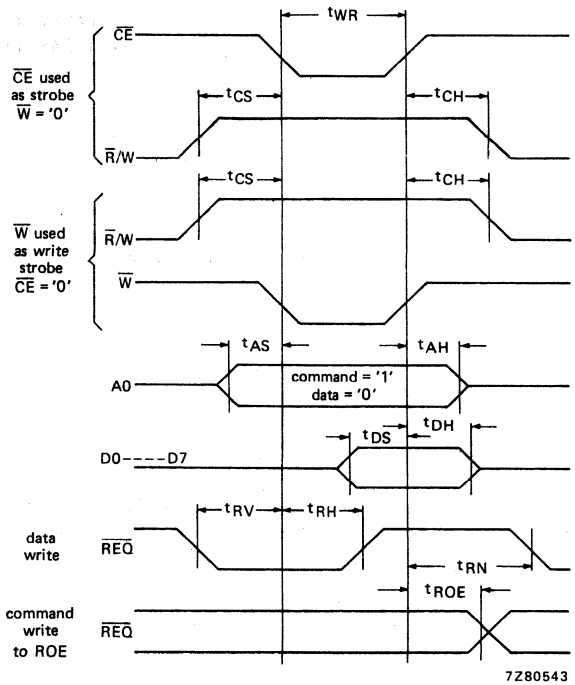
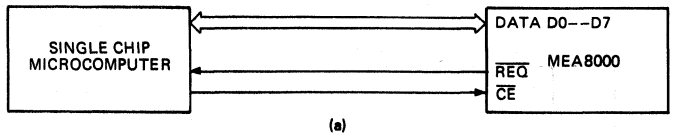
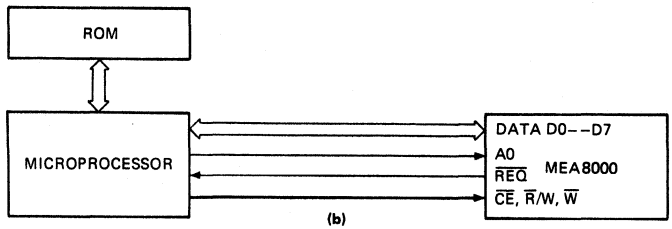


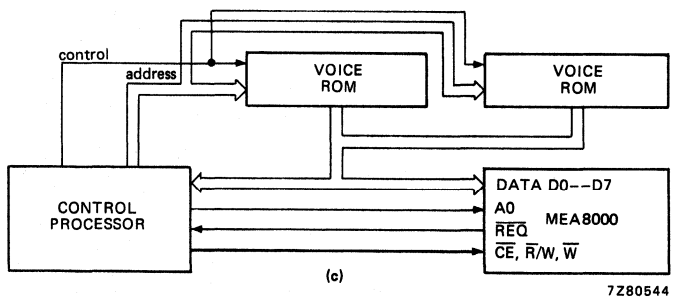
Fig. 7 Write timing.



(a) Minimum system of single chip microcomputer with voice ROM on board.



(b) MEA8000 as a microprocessor peripheral.



(c) Applications using separate voice ROMs.

Fig. 8 Typical applications.

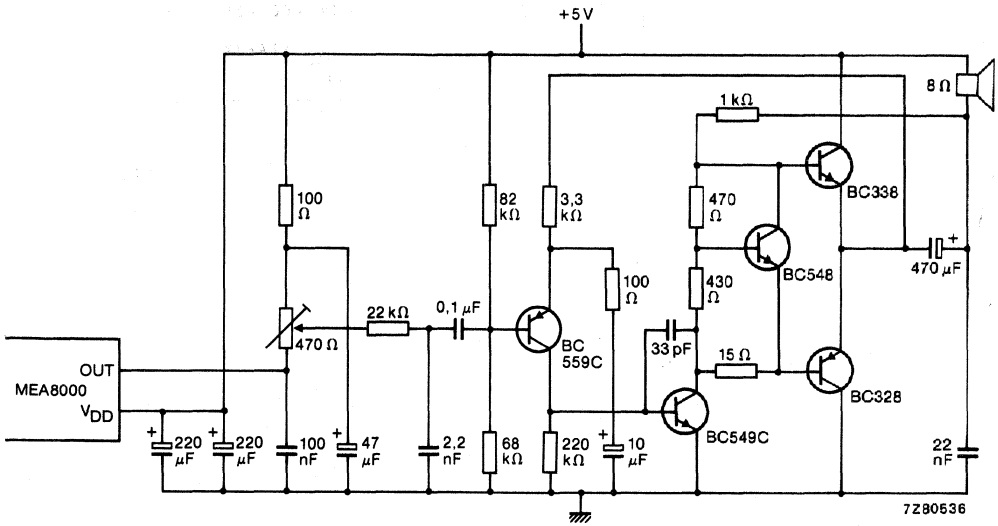


Fig. 9 Typical output applications.

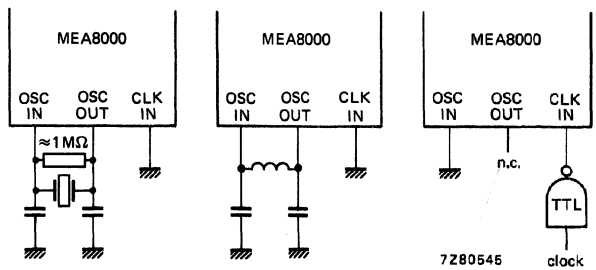


Fig. 10 Oscillator/clock configurations.

VOICE SYNTHESIZER

GENERAL DESCRIPTION

The PCF8200 is a CMOS integrated circuit for generating good quality speech from digital code with a programmable bit rate. The circuit is primarily intended for applications in microprocessor controlled systems, where the speech code is stored separately.
Applications include automotive, telephony, personal computers, annunciators, aids for the handicapped, and general industrial devices.

Features

- Male and female speech with good quality
- Speech-band from 0 to 5 kHz
- Bit-rate between 455 bits/second and 4545 bits/second
- Programmable frame duration
- Programmable speaking speed
- CMOS technology
- Operating temperature range −40 to + 85 °C
- Single 5 V supply with low power consumption and power-down stand-by mode
- Interfaces easily with most popular microcomputers and microprocessors through 8 bit parallel bus or I^2C bus
- Software readable status word (parallel bus or I^2C bus)
- BUSY-signal and REQN-signal hardware readable
- Internal low-pass filter and 11-bit D/A converter

QUICK REFERENCE DATA

parameter	symbol	min.	typ.	max.	unit
Supply voltage	V_{DD}	–	5	–	V
Supply current	I_{DD}	–	10	t.b.f.	mA
Supply current (stand-by)	$I_{DD(SB)}$	–	200	–	μA
Inputs					
Input voltage	V_{IH}	2,0	–	V_{DD}	V
Input voltage	V_{IL}	0	–	0,8	V
Input capacitance	C_I	–	7	–	pF
Outputs (D5 to D7)					
Output voltage high	V_{OH}	3,5	–	V_{DD}	V
Output voltage low	V_{OL}	0	–	0,4	V
Load capacitance	C_L	–	–	80	pF
Operating ambient temperature range	T_{amb}	−40	–	+ 85	°C

PACKAGE OUTLINE

24-lead DIL; plastic (SOT-101A).

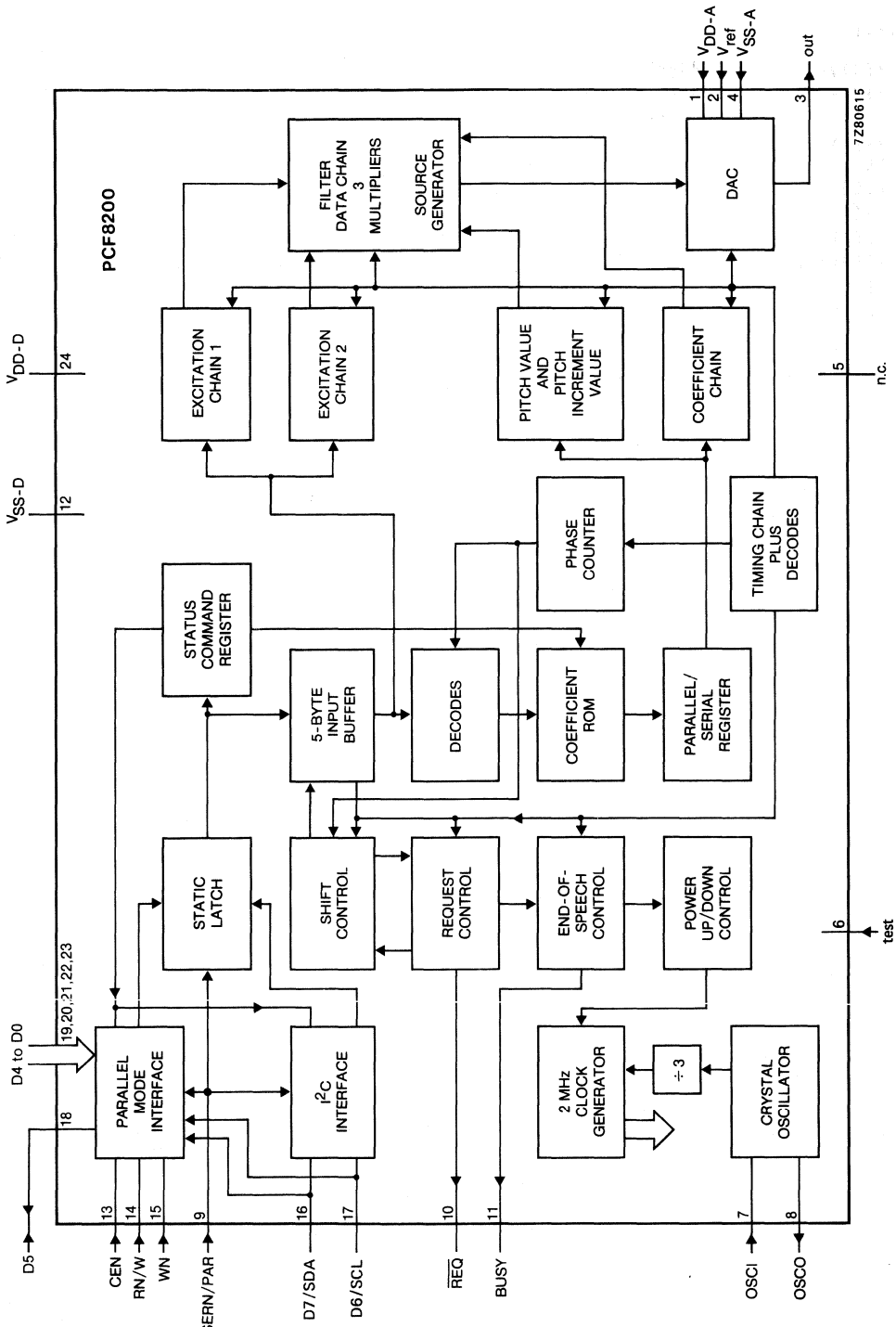


Fig. 1 Block diagram.

DEVELOPMENT DATA

PINNING

1	V_{DD-A}	supply
2	V_{REF}	supply
3	OUT	output
4	V_{SS-A}	supply
5	NC	not connected
6	TEST	input
7	OSCI	input
8	OSCO	output
9	SERN/PAR	input
10	REQN	output
11	BUSY	output
12	V_{SS-D}	supply
13	CEN	input
14	RN/W	input
15	WN	input
16	SDA/D7	input/output
17	SCL/D6	input/output
18	D5	input/output
19	D4	input
20	D3	input
21	D2	input
22	D1	input
23	D0	input
24	V_{DD-D}	supply

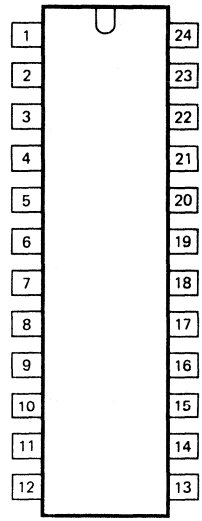


Fig. 2 Pinning diagram.

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Supply voltage*	V_{DD}	min.	−0,3	max.	7,5 V
Input voltage*	V_I	min.	−0,3	max.	7,5 V
Output voltage*	V_O	min.	−0,3	max.	7,5 V
Operating ambient temperature range	T_{amb}				−40 to + 85 °C
Storage temperature range	T_{stg}				−55 to + 125 °C

* Any pin with respect to V_{SS}.

FUNCTIONAL DESCRIPTION

The synthesizer has been designed for a vocal tract modelling technique of voice synthesis. An excitation signal is fed to a series of resonators. Each resonator simulates one of the formants in the original speech. It is controlled by two parameters, one for the resonant frequency and one for the bandwidth. Five formants are needed for male speech and four for female speech. The output of this system is defined by the excitation signal, the amplitude values and the resonator settings. By periodic updating of all parameters very high quality speech can be produced.

OPERATION

Speech characteristics change quite slowly, therefore the control parameters for the speech synthesizer can be adequately updated every few tens of milliseconds with interpolation during the interval to ensure a smooth changeover from one parameter value to the next. In the PCF8200 the standard-frame duration can be set to 8,8 , 10,4, 12,8 or 17,6 milliseconds with the speed-option, speaking speed, in the command-register.
The duration of each individual speech frame is programmable to be 1, 2, 3 or 5 times the standard-frame duration.

FD1, FD0	10	01	00	11	FS1, FS0
00	8,8	10,4	12,8	17,6	ms
01	17,6	20,8	25,6	35,2	ms
10	26,4	31,2	38,4	52,8	ms
11	44,0	52,0	64,0	88,0	ms

Table 1. Frame duration as a function of speed-option (FS1, FS0) and frame-duration (FD1, FD0).

The excitation signal is a random noise source for unvoiced sounds and a programmable pulse generator for voiced sounds. Both sources have an amplitude modulator which is updated 8 times in one speech-frame by linear interpolation. The pitch is updated every 1/8 of a standard frame.

The excitation signal is filtered with a five formant filter for male speech and a four formant filter for female speech. The formant filter is a cascade of all second-order sections. The control parameters, formant-frequency and formant-bandwidth, are updated eight times per speech frame by linear interpolation.
A block diagram of the formant synthesizer is shown in Fig. 3.

The filter output is upsampled to 80 kHz and filtered with a digital low-pass filter. Before the signal is digital to analogue converted (DAC), with an 11-bit switched capacitor DAC, the signal is multiplied with a DAC-amplitude factor. The use of a digital filter means that no external audio filtering is required for low-medium applications and minimal filtering is required for those applications requiring very high quality speech.

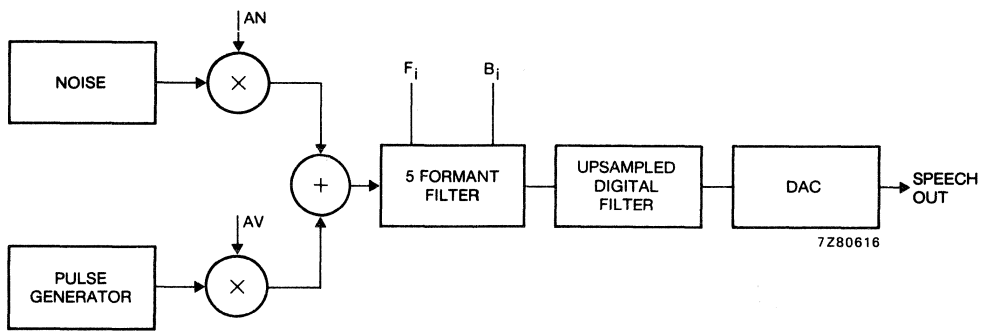


Fig. 3 Block diagram of formant synthesizer.

DATA FORMAT

Three types of format are used for data transfer to the synthesizer.

DAC-amplitude factor

The DAC-amplitude factor is one byte, which is used to optimize the digital speech signal to the 11-bit DAC. It is the first byte after a STOP or a BADSTOP or V_{DD} on. Table 2 indicates the amplitude factor.

byte	factor	dB
01110000	3,5	10,88
10110000	3,25	10,24
00110000	3,0	9,54
11010000	2,75	8,97
01010000	2,5	7,96
10010000	2,25	7,04
00010000	2,0	6,02
11100000	1,75	4,86
01100000	1,5	3,52
10100000	1,25	1,94
00100000	1,0	0,00
11000000	0,75	−2,50
01000000	0,5	−6,02
10000000	0,25	−12,04
00000000	0,0	
11110000	HEX code F0 and is not allowed as a DAC amplitude	

Table 2 DAC amplitude factor.

Start pitch

The second byte after a STOP or BADSTOP, or V_{DD} on is the start pitch. It is a one byte start value for the on-chip pitch-period generator.

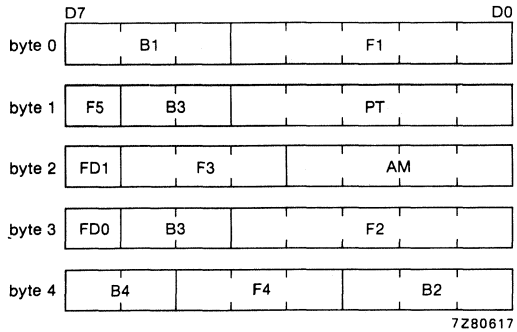
Frame Data

The frame data is a five byte block which contains the filter and source information:

pitch increment/decrement value	5 bits
amplitude	4 bits
frame duration	2 bits
frequency of 1st formant	5 bits
frequency of 2nd formant	5 bits
frequency of 3rd formant	3 bits
frequency of 4th formant	3 bits
frequency of 5th formant	1 bit
bandwidth of 1st formant	3 bits
bandwidth of 2nd formant	3 bits
bandwidth of 3rd formant	2 bits
bandwidth of 4th formant	2 bits
bandwidth of 5th formant	2 bits
	40 bits = 5 bytes

The frame-data bits are organized as shown in Fig. 4.

DEVELOPMENT DATA



It is not allowed to set byte 0 to the hexadecimal value E0.

Fig. 4 Format of frame-date.

CONTROL FORMAT

Command Write

A command write consists of two bytes, and it may occur before a data block. The four bits which can be written are shown in Fig. 5.

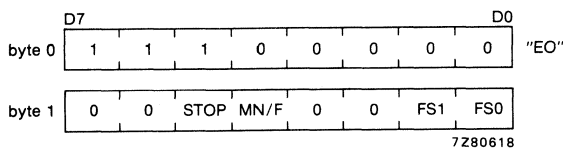


Fig. 5 Control write: first byte fixed, second byte control.

FS0, FS1 speed option

FS1	FS0	speech speed	standard-frame duration
0	0	100%	12,8 ms
0	1	123%	10,4 ms
1	0	145%	8,8 ms
1	1	73%	17,6 ms

MN/F, male/female option

MN/F = 0 male quantization table
= 1 female quantization table

STOP

STOP = 1 stop; repeat last complete frame with amplitude = 0 (no excitation signal)
= 0 if the frame data is not sent within the duration of a half frame, there will be a BADSTOP:

1. REQN = 1 STOP = 0
2. Repeat last frame with amplitude = 0
3. BUSY = 0

Status Read

Three status bits can be read out at any time without a preceding byte (E0).
This is shown in Fig. 6.

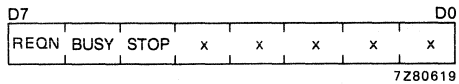


Fig. 6 Status read.

REQN = 1 No data required
= 0 Synthesizer requesting for new data
BUSY = 1 Busy (an utterance is pronounced)
= 0 Idle, REQN will set to 1; the synthesizer is in STOP or BADSTOP mode
STOP The STOP bit is the same as the stop bit written to the synthesizer during a command write.
STOP = 1, BUSY = 0 stopped by the user.
STOP = 0, BUSY = 0 BADSTOP because the data was not sent in time.

After initial power-up the status/command register is set to the following status:

FS0, FS1 = 0 Standard-frame duration of 12,8 ms
MN/F = 0 Male quantization table
STOP = 1
BUSY = 0 Idle
REQN = 1 No data required

DEVELOPMENT DATA

INTERFACE PROTOCOL

Data can be written to the synthesizer when REQN = 0 or, when REQN = 1 and BUSY = 0. Figure 7 shows the interface protocol of the synthesizer.

In parallel mode the synthesizer is activated by sending the DAC-amplitude factor. In serial mode the DAC-amplitude factor can be sent as soon as the synthesizer is powered-up.

The I²C transmitter/receiver will then acknowledge. When the request for the pitch-byte occurs the byte must be provided within the duration of a half standard frame. If the byte is not provided in time a BADSTOP will be generated.

During each data write operation, the status bit REQN will be set to '1'.
Within a frame data block, it disappears within a few microseconds, asking for the next byte of that block. If the bytes of frame data are not provided within the time-duration of a half frame, a BADSTOP will be generated.

I²C ADDRESS

On chip there is a I²C slave receiver/transmitter with the address:

7 6 5 4 3 2 1 0
0 0 1 0 0 0 0 R/W

POWER UP

The synthesizer will be set to power-up on a parallel-write sequence.

PAR-mode: The input-latches are active so they can receive the first byte

SER-mode: The I^2C transmitter/receiver will not acknowledge until the synthesizer has powered-up. To power up the synthesizer a parallel write sequence (Fig. 9) must be made to the synthesizer by using external logic for the control lines; at least one line must be toggled, CEN, while WN = 0 and RN/W = 1.
The synthesizer can be set to permanent power-up by hard-wired control pins (CEN = 0, RN/W = 1, WN = 0).

POWER DOWN MODE

When BUSY = 0 the synthesizer will be set to power-down. In the power-down mode the status/command register will be retained.
In power-down mode the clock-oscillator is switched off. After initial V_{DD} the synthesizer is in power-down mode.

SERN/PAR

SERN/PAR is hard-wired to V_{DD} or V_{SS}.

HANDLING

All inputs and outputs are protected against electrostatic charge under normal handling conditions.

DEVELOPMENT DATA

CHARACTERISTICS

T_{amb} = −45 to + 85 °C; supply voltage (V_{DD} to V_{SS}) = 4,5 V to 5,5 V with respect to V_{SS}, otherwise specified

parameter	symbol	min.	typ.	max.	unit
Supply					
Supply voltage	V_{DD}	4,5	5,0	t.b.f.	V
Supply current	I_{DD}	–	10	–	mA
Standby current	$I_{DD(SB)}$	–	200	–	μA
Inputs					
CEN, RN/W, WN, OSCI					
Input voltage HIGH	V_{IH}	2,0	–	V_{DD}	V
Input voltage LOW	V_{IL}	0	–	0,8	V
Input leakage current V_{in} = 0 to 5,5 V	I_{IR}	−10	–	10	μA
Rise and fall times (note 2)	t_{rf}	–	–	50	ns
Input capacitance	C_I	–	–	7	pF
PARALLEL MODE					
Input Characteristics (D0 to D7)					
Input voltage HIGH	V_{IH}	2,0	–	V_{DD}	V
Input voltage LOW	V_{IL}	0	–	0,8	V
Input leakage current (V_{in} = 0 to 5,5 V, output off)	I_{IR}	−10	–	10	μA
Input capacitance	C_I	–	–	7	pF
Output Characteristics (D5 to D7 only)					
Output voltage HIGH (I_{OH} = −100 μA)	V_{OH}	3,5	–	V_{DD}	V
Output voltage LOW (I_{OL} = 3,2 mA)	V_{OL}	0	–	0,4	V
Load capacitance	C_L	–	–	80	pF
Rise and fall times (note 3)	t_{rf}	–	–	50	ns
SERIAL MODE					
Input Characteristics (SDA and SDL)					
Input voltage HIGH	V_{IH}	3,0	–	V_{DD}	V
Input voltage LOW	V_{IL}	0	–	1,5	V
Input leakage current (V_{in} = 0 to 5,5 V, output off)	I_{IR}	−10	–	10	μA
Input capacitance	C_I	–	–	10	pF
Output Characteristics (SDA only, open drain)					
Output voltage LOW (I_{OL} = 3 mA)	V_{OL}	0	–	0,4	V

parameter	symbol	min.	typ.	max.	unit
OSCILLATOR					
Crystal frequency	f_{XTAL}	t.b.f.	6	t.b.f.	MHz
V_{REF}					
Reference voltage	V_{REF}	1,9	–	$\frac{V_{DD}-1,5}{1,25}$	V
Input leakage current	I_{IR}	–	t.b.f.		
Outputs					
REQN, BUSY					
Output voltage HIGH (I_{OH} = 100 μA)	V_{OH}	3,5	–	V_{DD}	V
Output voltage LOW (I_{QL} = 3,2 mA)	V_{OL}	0	–	0,4	V
Load capacitance	C_L	–	–	80	pF
Rise and fall times (note 3)	t_{rf}	–	–	50	ns
OUT					
Output voltage	V_{OUT}	0,66 x V_{REF}		1,34 x V_{REF}	V
Minimum external load		600	–	–	Ω
Timing characteristics (note 1) (Figs 8 and 9)					
Write enable	t_{WR}	200	–	–	ns
Data set-up for write	t_{DS}	150	–	–	ns
Data hold for write	t_{DH}	30	–	–	ns
Read enable	t_{RD}	200	–	–	ns
Data delay for read (note 2)	t_{DD}	–	–	150	ns
Data floating for read (note 2)	t_{DF}	–	–	150	ns
Control set-up	t_{CS}	0	–	–	ns
Control hold	t_{CH}	0	–	–	ns
REQ new (new byte of the same speech frame)	t_{RN}	–	t.b.f. ($\approx$ 3)		us
REQ Valid	t_{RV}	0	–	–	ns
REQ Hold	t_{RH}	–	250	t.b.f.	ns

NOTES TO THE CHARACTERISTICS

1. Timing reference level is 1,5 V; supply 5 V ± 10%; temperature range of −40 °C to 85 °C.
2. Levels greater than 2 V for a '1' or less than 0,8 V for a '0' are reached with a load of one TTL input and 50 pF.
3. Rise and fall times between 0,6 V and 2,2 V levels.

DEVELOPMENT DATA

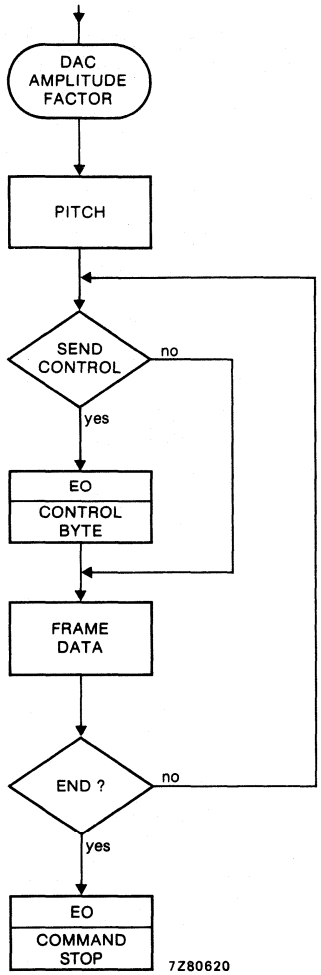
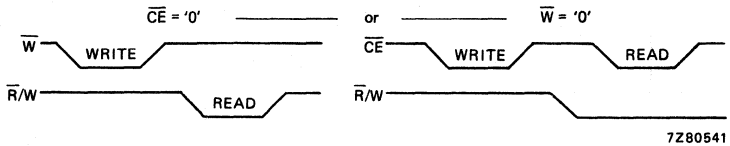


Fig. 7 Interface protocol.

Timing diagrams

The control signals CE, R/W and W have been specified to enable easy interface to most microprocessors and microcomputers. For instance with connection to an MAB8048 microcomputer the R/W and W inputs can be used as the RD and WR strobe inputs.



Typical connection of control signals.

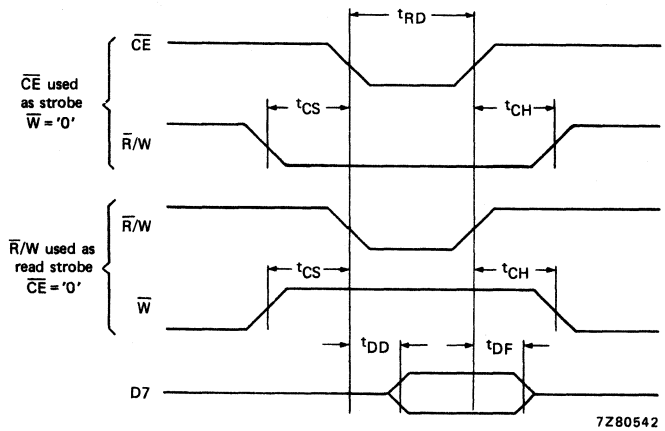


Fig. 8 Read timing.

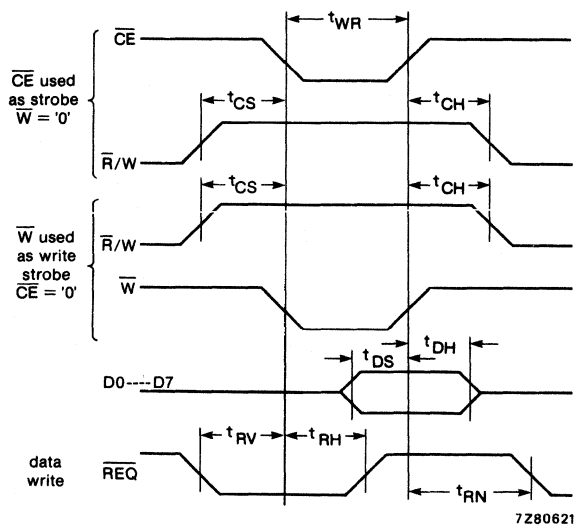


Fig. 9 Write timing.

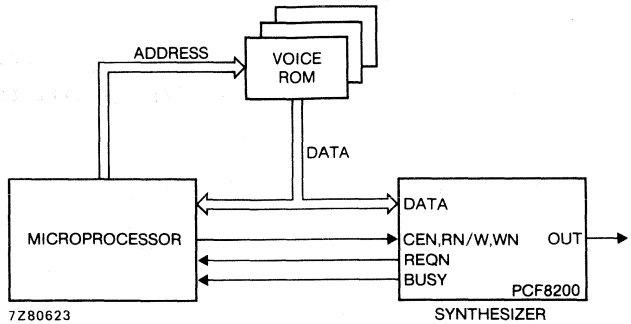


Fig. 10 Typical application configuration with parallel interface.

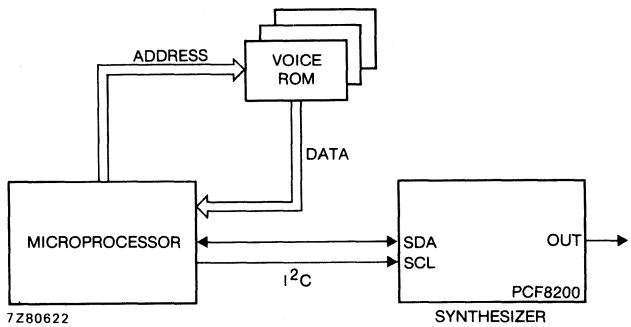


Fig. 11 Typical application configuration with series interface.

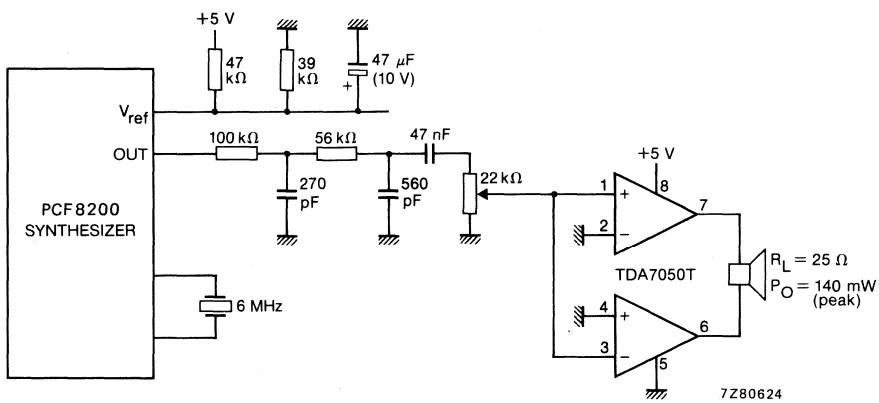


Fig. 12 An example of an output configuration.

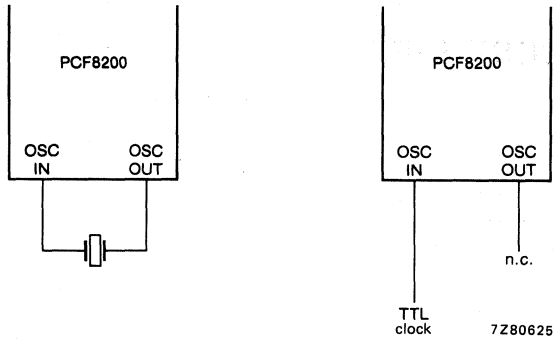


Fig. 13 Oscillator clock configurations.

LOW COST SPEECH DEMONSTRATION BOARD

GENERAL DESCRIPTION

The low cost speech demonstration board is designed to add voice output to existing card based electronic equipment with the minimum of additional effort and components. The majority of components used are of the CMOS type with low power consumption making the board suitable for battery operation.

Applications include speech evaluation and speech demonstration.

FEATURES

- PCF8200 speech synthesizer
  - Male and female speech of very high quality
  - CMOS technology
  - Extended operating temperature range
  - Programmable speaking speed
- Low current consumption
  - All major components use CMOS technology (PCF8200, 80C39 and 27C64)
- Very large vocabulary up to 12 minutes
  - 4 EPROM sockets
  - EPROM selection for 27C16 to 27C256
  - Low data rates for synthesizer (average 1500 bits per second)
- Easy interfacing
  - 8-bit parallel data bus/key switch input
  - Volume control, speaker connection
  - Control signals (e.g. RESET, BUSY etc etc)
- Simple operating modes
  - ROM selection
  - Word sequence within a ROM
  - Repeat last utterence
  - Control software is readily customizeable
  - To implement parameter download from external source
- Single Eurocard size PC board
- Single + 5 V supply
- Low cost

APPLICATIONS

- OEM design-in
- May be simply used with many card systems for speech evaluation
- Speech demonstration
  - Particularly simple when used with the OM8201 (Speech Demonstration Box)

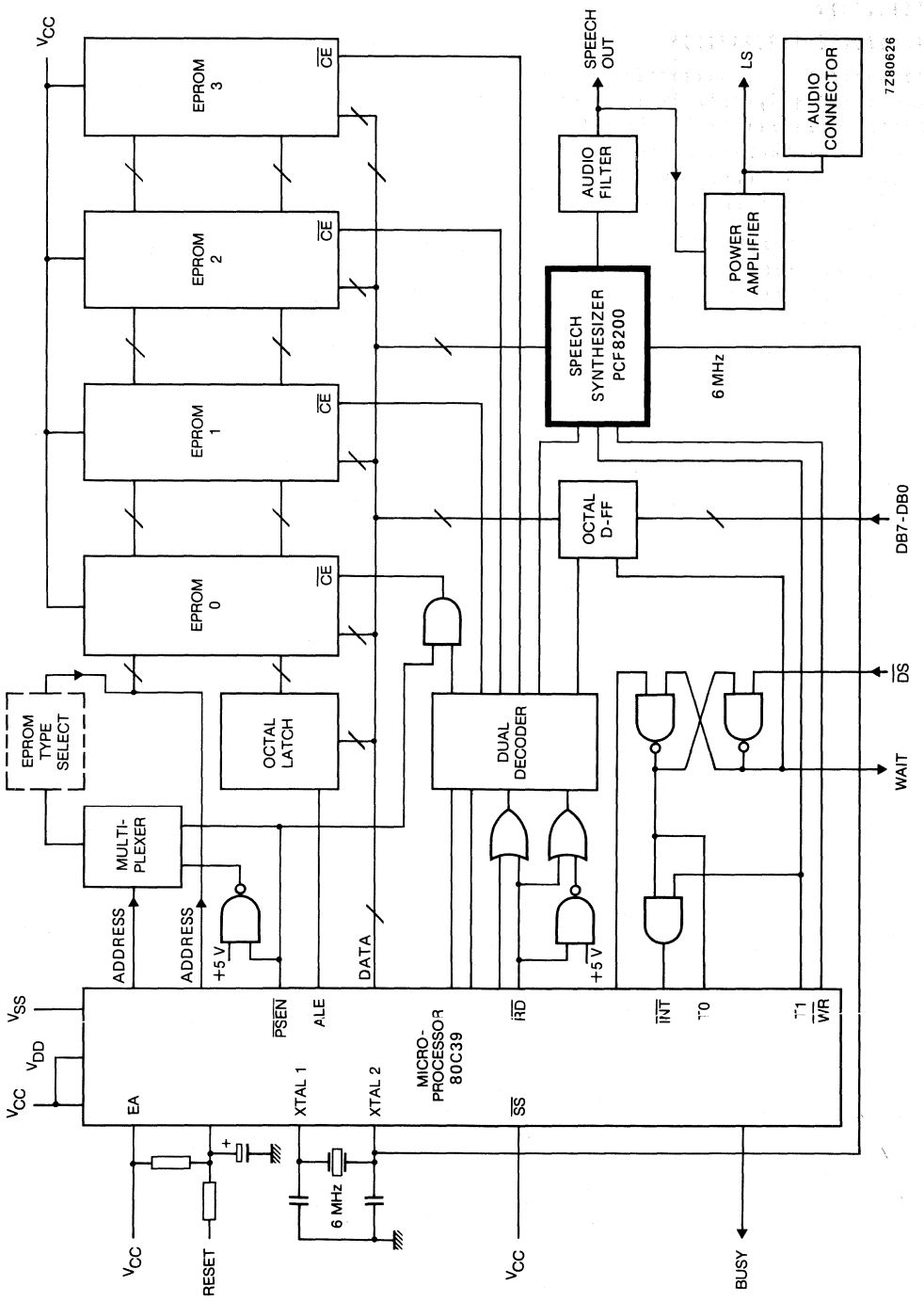


Fig. 1 Block diagram.

OPERATION

HARDWARE DESCRIPTION

The main controlling microprocessor is an 80C39 running at 6 MHz. This device supplies all of the main controlling signals for the board operation and the interfacing to any external system.
Four sockets are provided for EPROMS which contain speech coding. These may be 27C16 types, through to 27C256 types; the sockets will be a low insertion force type to allow for easy customizing.
The board will be supplied with one socket occupied by a 27C64 which will contain the control program and some speech examples. All four EPROM sockets must contain the same EPROM type.

The speech synthesizer PCF8200 converts the coding into a speech output. This synthesizer has been designed to simulate the human vocal tract using five formants for male and four formants for female speech. Periodic updating of the parameters for these formants can produce very high quality speech.

The output of the synthesizer can be fed into an audio amplifier, TDA7050, via a resistor-capacitor filter network which provides a frequency cut-off above 5 kHz of about 25 dB. The configuration of the audio amplifier used on this board gives an output of 140 mW peak power into a 25 Ω speaker from a 5 V supply.

Connections are made to the board via a standard DIN/IEC connector. This allows access to the 8-bit parallel data bus so that speech coding from an external source may be used, if implemented, and allows the selection of speech phrases by an external system, such as a microcomputer or even a bank of switches. The same connector also permits the addition of a volume control, loudspeaker, a high impedance audio output, and power supply. The control signals RESET, BUSY, WAIT and DS are also taken to the outside of the board. There is also a loudspeaker plug on the board.

All components are contained on a standard single Eurocard, and therefore suitable for rack mounted equipment.

SOFTWARE DESCRIPTION

All the software required to operate the board is contained in the only EPROM supplied. The software is written in modular from so that it is possible for a customer to alter or add to any particular function which suits his applications. An industrial standard microprocessor was chosen so that readily available development systems could be used to facilitate this modification.

There are four main modes of operation:

- ROM Selection
- Word Sequence
- Repeat Word
- Speaking Speed Selection

These modes are all controlled by software.

ROM Selection mode permits access to an individual EPROM and pronounces the first utterence from that EPROM.

Word Sequence gives the next word (activated by repeated access to the same EPROM) and if continually exercised will keep looping on the words in that EPROM.

The Repeat Word command allows indefinate repetition of the last utterance pronounced.

The Speaking Speed Selection allows the utterence to be pronounced at a different speed.

The software also controls the address sequencing within the utterance and ensures that the required data is supplied to the synthesizer.

There are also some examples of words/utterences encoded in the remainder of the supplied EPROM. These words are intended for demonstration purposes and will show the features of the synthesizer when selected. The main features being illustrated are:

- Male speech in several languages
- Female speech in several languages
- Programmable speaking speed

ORDERING INFORMATION

Product name: Low Cost Speech Demonstration Board

Type number: OM8200

Ordering code: 9337 541 30000

Orders should be placed with your local Philips/Signetics agency.

SPEECH DEMONSTRATION BOX

GENERAL DESCRIPTION

Speech demonstration box OM8201 is designed to be used in conjunction with the low cost speech demonstration board OM8200. The box contains all the necessary components to drive the board. The combination of these two components make an extremely attractive demonstration unit.

FEATURES

- Low cost
- Can use unmodified OM8200 board which allows access to all features of the OM8200
- Single + 9 V supply
  - Low power consumption therefore permits battery operation
  - External power supplies may also be used
  - Voltage is regulated and dropped to a standard + 5 V for the OM8200 board
- Simple mechanical construction
  - Allows easy access to the OM8200 for changing EPROMS
- Contains all peripherals needed to drive the OM8200

HARDWARE DESCRIPTION

The box contains a set of eight keypad switches which are connected to the data bus. Four switches can select which EPROM your speech data is derived from. Repeated pressing of an EPROM switch increments the expression number which will be uttered. To repeat the last expression, a separate switch must be activated.

It is possible in the PCF8200 to change the rate of speaking to 73%, 123% or 145% of the normal speed. A switch has been included on the box which will sequence through the speed options making the same utterance every time.

One of the two remaining switches is the master reset for the program and the other is for future enhancements of the box.

Included in the box are, the volume control for the amplifier, the loudspeaker, and a high impedance audio output.

The final piece of electronics is the power supply. This can be supplied from a + 9 V internal battery or from a + 9 V external supply. The + 9 V is regulated to a + 5 V supply which is then fed to other parts of the box and to the OM8200.

The box is of simple construction and allows easy access to the OM8200 for changing of EPROMS.

SOFTWARE DESCRIPTION

There is no software in the OM8201. The software of the OM8200 may be used in an unmodified form without any problems. However, if changes have been made to the control program of the OM8200 then different functions for the switches of the box can be achieved.

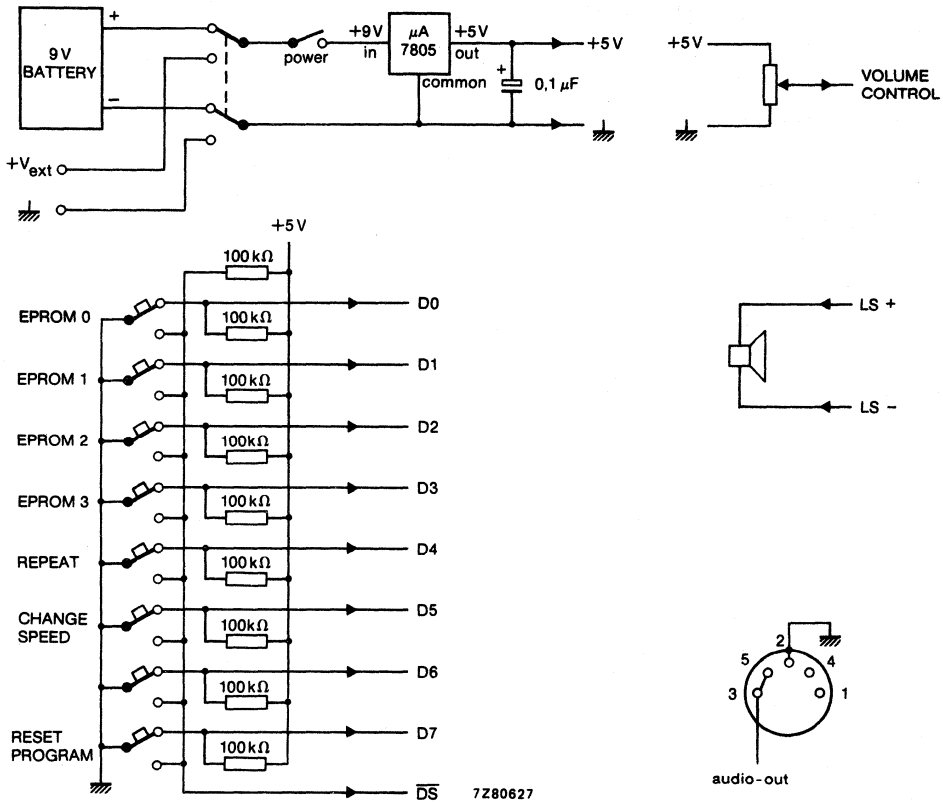


Fig. 1 Schematic diagram.

ORDERING INFORMATION

Product name: Speech Demonstration Box

Type number: OM8201

Ordering code: 9337 541 40000

N.B. OM8200 must be ordered as well if this box is to be used in demonstration mode.
The order number for the OM8200 is 9337 541 30000.

Orders should be placed with your local Philips/Signetics agent.

SPEECH ANALYSIS/EDITING SYSTEM

GENERAL DESCRIPTION

The OM8210 is a speech analysing/editing system, and comprises of a speech adapter box and associated software. The system uses the HP9816S personal computer.
The OM8210 and the HP9816S function together to produce speech coding for the PCF8200.
The system has many commands available, mostly single key operations, which gives it flexability.

FEATURES

- Input sampling of analogue speech signals
- Speech analysis
- Graphic parameter representation
- Parameter editing screen
- Conversion of parameters to PCF8200 synthesizer
- EPROM programming
- Parameter storage on floppy disc
- Speech output via PCF8200 voice synthesizer

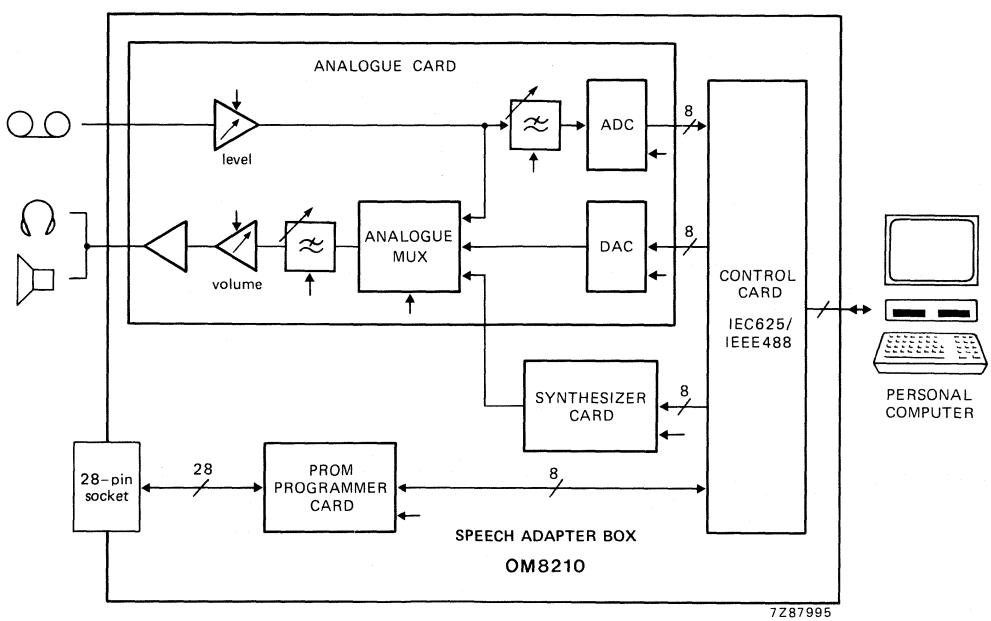


Fig. 1 Block diagram.

HARDWARE DESCRIPTION

The hardware for the OM8210 is contained in an attractive box with access to all the interconnections (IEC 625, interface loudspeaker, headphones, tape input, and EPROM socket), from the front panel. There are four single Eurocards and a power supply forming the speech adapter box.
These cards are:

- Analogue Card
- Synthesizer Card
- EPROM Card
- Control Card

Analogue Card

On this card, the level of the recorded audio input signal is adjusted by an electronic potentiometer. Before the audio is sampled, frequencies higher than half the sampling frequency are removed by a switched capacitor filter of the type normally used for codecs. A 12-bit analogue-to-digital converter (ADC) produces the digital samples that are sent to the control card. An 8-bit digital-to-analogue converter (DAC) on the analogue card allows the sampled speech to be output. The audio input signal, the sampled speech and the synthesized speech are selected by an analogue multiplexer, filtered, and adjusted for volume before reproduction by a loudspeaker.

The use of integrated electronic potentiometers and codec filters substantially reduces the number of components required while maintaining high performance.

Synthesizer Card

This card accommodates the PCF8200 voice synthesizer and a small amount of peripheral components and a socket for the MEA8000 voice synthesizer.

EPROM Programmer Card

This card allows four different types of EPROM (2716, 2732, 2732A and 2764) to be programmed under software control. All the hardware to generate the programming voltages and the programming waveforms are on this card.

Control Card

This card performs three functions:

- IEC 625/IEEE 488 interface
- Control sequencer
- Clock generator

The IEC/IEEE interface is a simple talker/listener implementation with a HEF4738 circuit.

An FPLA control sequencer provides the handshake signals for IEC/IEEE interface and the chip enable signals for the rest of the system (the ADC, the DAC, the synthesizer and control circuits).

The filter sampling frequency is generated with a software programmable PLL frequency synthesizer. The speech sampling frequency is derived from the filter sampling frequency by frequency division. Hence, the filter frequency cut-off and the sample rate of the ADC and the DAC are automatically linked.

The hardware includes all the necessary cables, adapter plug, loudspeaker, headphone and power supply.

SOFTWARE DESCRIPTION

The software for this speech coding system has been developed and arranged for optimum user convenience. There are eight modes available.

Each mode and each command in the mode is selected by single key entries. Commands that can destroy data have to be confirmed before they are executed. More than 100 commands are available. The modes are:

Sample Mode	Samples and digitizes the recorded speech, the amplitude can be checked and speech segments selected. The sampled speech is stored in a memory and can be displayed or made audible.
Analysis Mode	Generates speech parameters from samples. The analysis selects the voiced/unvoiced sections, extracts the formants (5 for male and 4 for female), amplitude, and the pitch, and quantisizes the speech parameters.
Parameter Edit Mode	Speech parameters are displayed graphically on the VDU and can be edited to correct errors in the analysis, improve speech quality by altering contours, or amplitudes, concatenate sounds and optimize data rate by editing the frame duration.
Code Mode	Generates PCF8200 code and permits the arrangement of utterences in the optimum order of application. This mode also generates the address map at the head of the EPROM.
EPROM Mode	Used to program/read EPROMS with data for the code memory also possible is a new check, bit check and verification commands.
File Mode	Stores speech parameters or codes on disc, can also assemble code speech segment from an already existing library.
Media Mode	For diskette initialization and making back-up copies.
Option Mode	Allows the system configuration to be read or changed.

DEVELOPMENT DATA

The software is supplied on two diskettes, one labelled 'BOOT' which wakes up the system and also contains the system library routines. The other diskette labelled 'SPEECH' contains the speech program, the disc initialization and the file handler programs. The 'BOOT' disc is not required during operation, giving a free disc drive with the system for a diskette to store speech parameter files.

Computer System

The following equipment is required to make a complete editing system:

- HP9816S-630 (optimum computer type)
- HP9121D (dual floppy disc)
- Additional memory card for the HP9816S (512 k bytes total required)

ORDERING INFORMATION

Product name: Speech Analysis/Editing System

Type number: OM8210

Ordering code: 9337 561 50000

The Hewlett Packard computer should be purchased from your local agents.
The OM8210 should be ordered through your local Philips/Signetics agent.

MICROPROCESSOR CONTROLLED STEREO SOUND GENERATOR FOR SOUND EFFECTS AND MUSIC SYNTHESIS

GENERAL DESCRIPTION

The SAA1099 is a monolithic integrated circuit designed for generation of stereo sound effects and music synthesis.

Features

- Six frequency generators
  - eight octaves per generator
  - 256 tones per octave
- Two noise generators
- Six noise/frequency mixers
- Twelve amplitude controllers
- Two envelope controllers
- Two 6-channel mixers/current sink analogue output stages
- TTL input compatible
- Readily interfaces to 8-bit microcontroller
- Minimal peripheral components
- Simple output filtering

Applications

- Consumer games systems
- Home computers
- Electronic organs
- Arcade games
- Toys
- Chimes/alarm clocks

QUICK REFERENCE DATA

Supply voltage (pin 18)	V_{DD}	typ.	5 V
Supply current (pin 18)	I_{DD}	typ.	70 mA
Reference current (pin 6)	I_{ref}	typ.	250 μA
Total power dissipation	P_{tot}		500 mW
Operating ambient temperature range	T_{amb}		0 to + 70 °C

PACKAGE OUTLINE

18-lead DIL; plastic (SOT-102CS).

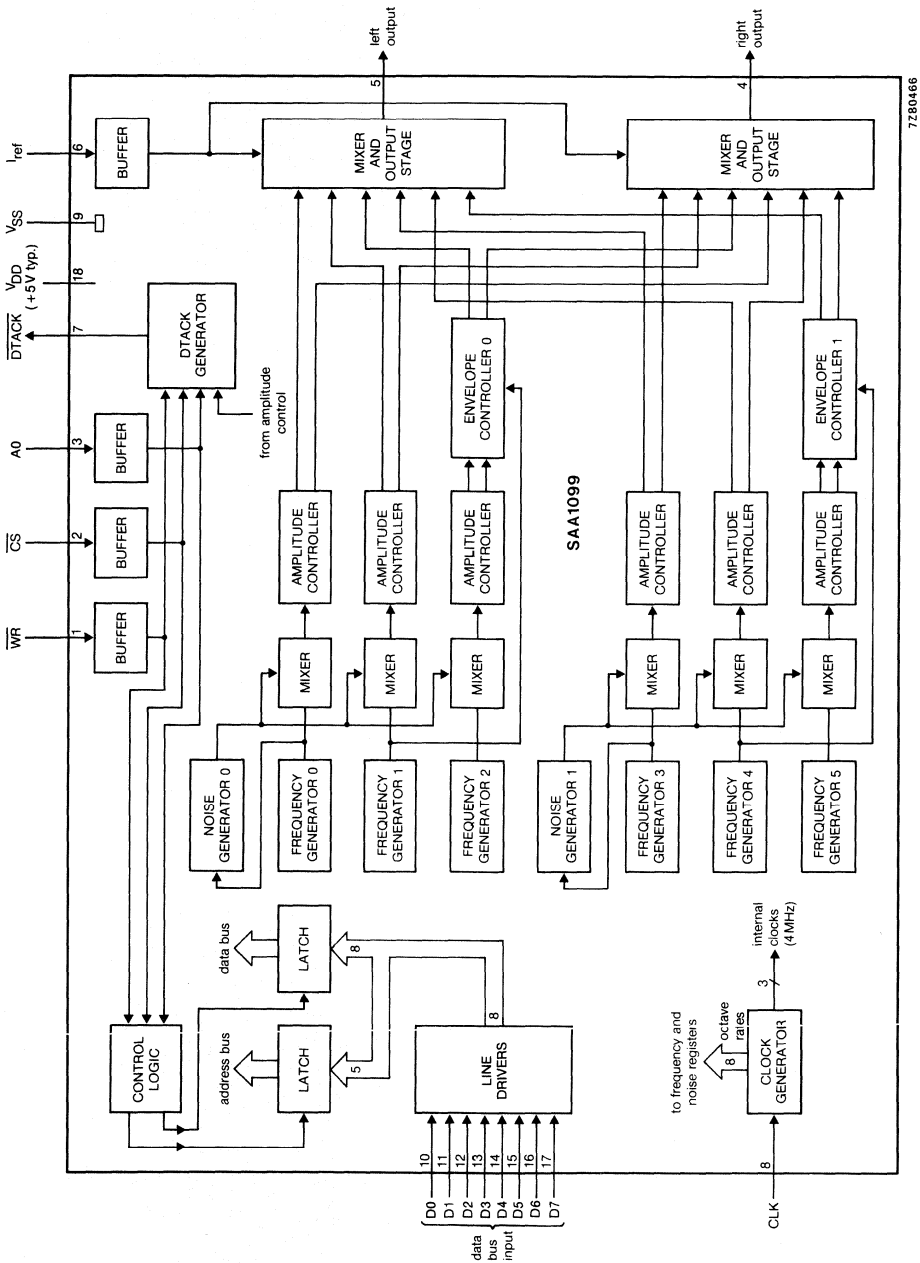


Fig. 1 Block diagram.

PINNING

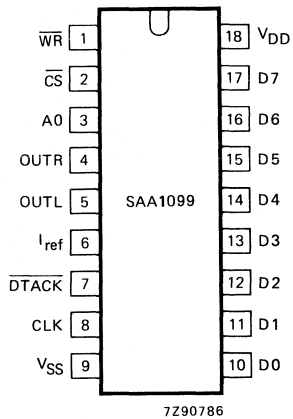


Fig. 2 Pinning diagram.

PIN DESIGNATION

DEVELOPMENT DATA

1	$\overline{WR}$	**Write Enable**: active LOW input which operates in conjunction with $\overline{CS}$ and A0 to allow writing to the internal registers.
2	$\overline{CS}$	**Chip Select**: active LOW input to identify valid $\overline{WR}$ inputs to the chip. This input also operates in conjunction with $\overline{WR}$ and A0 to allow writing to the internal registers.
3	A0	**Control/Address select**: input used in conjunction with $\overline{WR}$ and $\overline{CS}$ to load data to the control register (A0 = 0) or the address buffer (A0 = 1).
4	OUTR	**Right channel output**: a 7-level current sink analogue output for the 'right' component. This pin requires an external load resistor.
5	OUTL	**Left channel output**: a 7-level current sink analogue output for the 'left' component. This pin requires an external load resistor.
6	I_{ref}	**Reference current supply**: used to bias the current sink outputs.
7	$\overline{DTACK}$	**Data Transfer Acknowledge**: open drain output, active LOW to acknowledge successful data transfer. On completion of the cycle $\overline{DTACK}$ is set to inactive.
8	CLK	**Clock**: input for an externally generated clock at a nominal frequency of 8 MHz.
9	V_{SS}	**Ground**: 0 V.
10-17	D0-D7	**Data**: Data bus input.
18	V_{DD}	**Power supply**: + 5 V typical.

FUNCTIONAL DESCRIPTION

The following sections provide a detailed functional description of the SAA1099 as shown in the block diagram, Fig. 1.

Frequency generators

Six frequency generators can each select one of 8 octaves and one of 256 tones within an octave. A total frequency range of 31 Hz to 7,81 kHz is available. The outputs may also control noise or envelope generators. All frequency generators have an enable bit which switches them on and off, making it possible to preselect a tone and to make it inaudible when required. The frequency generators may be synchronized using the frequency reset bit.

The frequency ranges per octave are:

Octave	Frequency range
0	31 Hz to 61 Hz
1	61 Hz to 122 Hz
2	122 Hz to 244 Hz
3	245 Hz to 488 Hz
4	489 Hz to 977 Hz
5	978 Hz to 1,95 kHz
6	1,96 kHz to 3,91 kHz
7	3,91 kHz to 7,81 kHz

Noise generators

The two noise generators both have a programmable output. This may be a software controlled noise via one of the frequency controlled generators or one of three pre-defined noises. There is no tone produced by the frequency generator when it is controlling the noise generator. The noise produced is based on double the frequency generator output, i.e. a range of 61 Hz to 15,6 kHz.

In the event of a pre-defined noise being chosen, the output of noise generator 0 can be mixed with frequency generator 0, 1 and 2; and the output of noise generator 1 can be mixed with frequency generator 3, 4 and 5. In order to produce an equal level of noise and tone outputs (when both are mixed) the amplitude of the tone is increased. The three pre-defined noises are based on a clock frequency of 7,8 kHz, 15,6 kHz or 31,25 kHz.

Noise/frequency mixers

Six noise/frequency mixers each with four selections

- Channel off
- Frequency only
- Noise only
- Noise and frequency

Each mixer channel has one of the frequency generator outputs fed to it, three channels use noise generator 0 and the other three use noise generator 1.

Amplitude controllers

Each of the six channel outputs from the mixer is split up into a right and left component giving effectively twelve amplitude controllers. An amplitude of 16 possible levels is assigned to each of the twelve signals. With this configuration a stereo effect can be achieved by varying only the amplitude component. The moving of a sound from one channel to the other requires, per tone, only one update of the amplitude register contents.

When an envelope generator is used, the amplitude levels are restricted. The number of levels available is then reduced to eight. This is achieved by disabling the least significant bit (LSB) of the amplitude control.

Envelope controllers

Two of the six tone generators are under envelope control. This applies to both the left and right outputs from the tone generator.
The envelope has the following eight possible modes:

- Amplitude is zero
- Single attack
- Single decay
- Single attack-decay (triangular)
- Maximum amplitude
- Continuous attack
- Continuous decay
- Continuous attack-decay

The timing of the envelope controllers is programmable using one of the frequency generators (see Fig. 1). When the envelope mode is selected for a channel its control resolution is halved for that channel from 16 levels to 8 levels by rounding down to the nearest even level.
There is also the capability of controlling the 'right' component of the channel with inverse of the 'left' component, which remains as programmed.

A direct enable permits the start of an envelope to be defined, and also allows termination of an envelope at any time. The envelope rate may be controlled by a frequency channel (see Fig. 1), or by the microprocessor writing to the address buffer register. If the frequency channel controlled is OFF (NE = FE = 0) the envelope will appear at the output, which provides an alternative 'non-square' tone capability. In this event the frequency will be the envelope rate, which provided the rate is from the frequency channel, will be a maximum of 1 kHz. Higher frequencies of up to 2 kHz can be obtained by the envelope resolution being halved from 16 levels to 8 levels. Rates quoted are based on the input of a 8 MHz clock.

Six-channel mixers/current sink analogue output stages

Six channels are mixed together by the two mixers allowing each one to control one of six equally weighted current sinks, to provide a seven level analogue output.

Command/control select

In order to simplify the microprocessor interface the command and control information is multiplexed. To select a register in order to control frequencies, amplitudes, etc. the command-register has to be loaded. The contents of this register determines to which register the data is written in the next control-cycle. If a continuous update of the control-register is necessary, only the control-information has to be written (the command-information does not change).
If the command/control select (A0) is logic 0, the byte transfer is control; if A0 is logic 1, the byte transfer is command.

Interface to microprocessor

The SAA1099 is a data bus based I/O peripheral. Depending on the value of the command/control signal (A0) the $\overline{CS}$ and $\overline{WR}$ signals control the data transfer from the microprocessor to the SAA1099. The data-transfer-acknowledge ($\overline{DTACK}$) indicates that the data transfer is completed. When, during the write cycle, the microprocessor recognizes the $\overline{DTACK}$, the bus cycle will be completed by the processor.

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Supply voltage (pin 18)	V_{DD}	−0,3 to	+ 7,5	V
Maximum input voltage	V_I	−0,3 to	+ 7,5	V
at V_{DD} = 4,5 to 5,5 V	V_I	−0,5 to	+ 7,5	V
Maximum output current	I_O	max.	10	mA
Total power dissipation	P_{tot}		500	mW
Storage temperature range	T_{stg}	−55 to	+ 125	°C
Operating ambient temperature range	T_{amb}	0 to	+ 70	°C
Electrostatic handling*	V_{es}	−1000 to	+ 1000	V

* Equivalent to discharging a 250 μF capacitor through a 1 kΩ series resistor.

D.C. CHARACTERISTICS

V_{DD} = 5 V ± 10%; T_{amb} = 0 to 70 °C; unless otherwise specified

parameter	symbol	min.	typ.	max.	unit
Supply					
Supply voltage	V_{DD}	4,5	5,0	5,5	V
Supply current	I_{DD}	–	70	100	mA
Reference current (note 1)	I_{ref}	100	250	400	μA
INPUTS					
Input voltage HIGH	V_{IH}	2,0	–	6,0	V
Input voltage LOW	V_{IL}	−0,5	–	0,8	V
Input leakage current	$\pm I_{LI}$	–	–	10	μA
Input capacitance	C_I	–	–	10	pF
OUTPUTS					
$\overline{DTACK}$ (open drain; note 2)					
Output voltage LOW at I_{OL} = 3,2 mA	V_{OL}	0	–	0,4	V
Voltage on pin 7 (OFF state)	V_{7-9}	−0,3	–	6,0	V
Output capacitance (OFF state)	C_O	–	–	10	pF
Load capacitance	C_L	–	–	150	pF
Output leakage current (OFF state)	$-I_{LO}$	–	–	10	μA
Audio outputs (pins 4 and 5)					
With fixed I_{ref} (note 3)					
One channel on	I_{01}/I_{ref}	90	–	120	%
Six channels on	$I_{06}/6xI_{ref}$	85	–	110	%
With I_{ref} = 250 μA; R_L = 1,5 kΩ (± 5%)					
One channel on	I_{01}/I_{ref}	90	–	110	%
Six channels on	$I_{06}/6xI_{ref}$	85	–	105	%
Output current one channel on	I_{01}	225	–	275	μA
Output current six channels on	I_{06}	1,3	–	1,6	mA
With resistor supplying I_{ref} (note 4)					
Output current one channel on	I_{01}	150	–	350	μA
Output current six channels on	I_{06}	0,9	–	1,9	mA
Load resistance	R_L	600	–	–	Ω
D.C. leakage current all channels off	$-I_{LO}$	–	–	10	μA
Maximum current difference between left and right current sinks (note 5)	$\pm I_{Omax}$	–	–	15	%
Signal-to-noise ratio (note 6)	S/N	–	tbf	–	dB

DEVELOPMENT DATA

A.C. CHARACTERISTICS

V_{DD} = 5 V ± 10%; T_{amb} = 0 to 70 °C; timing measurements taken at 2,0 V for a logic 1 and 0,8 V for a logic 0 unless otherwise specified (see waveforms Figs 3 and 4)

parameter	symbol	min.	typ.	max.	unit
Bus interface timing (see Fig. 3)					
A0 set-up time to $\overline{CS}$ fall	t_{ASC}	0	–	–	ns
$\overline{CS}$ LOW to $\overline{WR}$ fall	t_{CSW}	30	–	–	ns
A0 set-up time to $\overline{WR}$ fall	t_{ASW}	50	–	–	ns
$\overline{WR}$ LOW time	t_{WL}	100	–	–	ns
Data bus valid to $\overline{WR}$ rise	t_{BSW}	100	–	–	ns
$\overline{DTACK}$ fall delay from $\overline{WR}$ fall (note 7)	t_{DFW}	0	–	85	ns
A0 hold time from $\overline{WR}$ HIGH	t_{AHW}	0	–	–	ns
$\overline{CS}$ hold time from $\overline{WR}$ HIGH	t_{CHW}	0	–	–	ns
Data bus hold time from $\overline{WR}$ HIGH	t_{DHW}	0	–	–	ns
$\overline{DTACK}$ rise delay from $\overline{WR}$ HIGH	t_{DRW}	0	–	100	ns
Bus cycle time (note 8)	t_{CY}	$4t_{CLK}$	–	–	
Bus cycle time (note 9)	t_{CY}	$16t_{CLK}$	–	–	
Clock input timing (see Fig. 4)					
Clock period	t_{CLK}	120	125	255	ns
Clock LOW time	t_{LOW}	55	–	–	ns
Clock HIGH time	t_{HIGH}	55	–	–	ns

Notes to the characteristics

1. Using an external constant current generator to provide a nominal I_{ref} or external resistor connected to V_{DD}.
2. This output is short-circuit protected to V_{DD} and V_{SS}.
3. Measured with I_{ref} a constant value between 100 and 400 μA; load resistance (R_L) allowed to match E12 (5%) in all applications via:

$$R_L = 0{,}6\ [I_{ref}]^{-1} - 16\ [I_{ref}]^{-0{,}5} \pm 12\%$$

4. Measured with R_{ref} = 10 kΩ (± 5%) connected between I_{ref} and V_{DD}; R_L = 1,5 kΩ (± 5%); OUTR and OUTL short-circuit protected to V_{SS}.
5. Left and right outputs must be driven with identical configuration.
6. Sample tested value only.
7. This timing parameter only applies when no wait states are required; otherwise parameter is invalid.
8. The minimum bus cycle time of four clock periods is for loading all registers except the amplitude registers.
9. The minimum bus cycle time of 16 clock periods is for loading the amplitude registers. In a system using $\overline{DTACK}$ it is possible to achieve minimum times of 500 ns. Without $\overline{DTACK}$ the parameter given must be used.

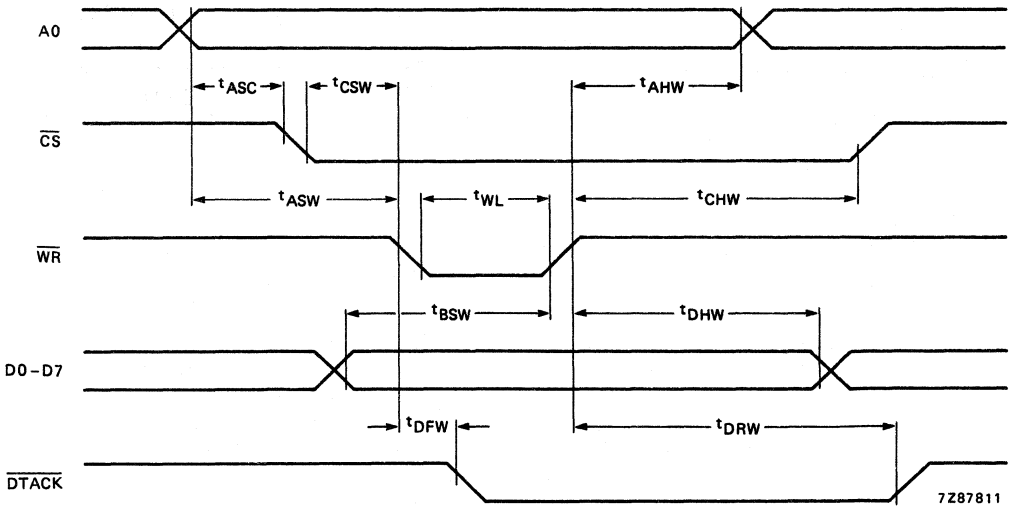


Fig. 3 Bus interface waveforms.

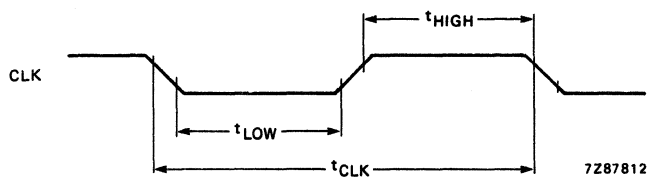


Fig. 4 Clock input waveform.

APPLICATION INFORMATION

Device operation

The SAA1099 uses pulse width modulation to achieve amplitude and envelope levels. The twelve signals are mixed in an analogue format (6 'left' and 6 'right') before leaving the chip. The amplitude and envelope signals chop the output at a minimum rate of 62,5 kHz, compared with the highest tone output of 7,81 kHz. Simple external low-pass filtering is used to remove the high frequency components.

Rates quoted are based on the input of a 8 MHz clock.

A data bus based write only structure is used to load the on-board registers. The data bus is used to load the address for a register, and subsequently the data to that register. Once the address is loaded multiple data loads to that register can be performed.

The selection of address or data is made by the single address bit A0, as shown in register maps Table 1 and Table 2.

The bus control signals $\overline{WR}$ and $\overline{CS}$ are designed to be compatible with a wide range of microprocessors, a $\overline{DTACK}$ output is included to optimise the interface with an S68000 series microprocessor. In most bus cycles $\overline{DTACK}$ will be returned immediately, this applies to all register address load cycles and all except amplitude data load cycles. With respect to amplitude data, a number of wait cycles may need to be performed, depending on the time since the previous amplitude load. $\overline{DTACK}$ will indicate the number of required waits.

Register description (see Tables 2 and 3)

The amplitudes are assigned with 'left' and 'right' components in the same byte, on a channel by channel basis. The spare locations that are left between blocks of registers is to allow for future expansion, and should be written as zero's. The tone within an octave is defined by eight bits and the octave by three bits. Note that octaves are paired (0/1, 2/3 etc.). The frequency and noise enables are grouped together for ease of programming. The controls for noise 'colour' (clock rate) are grouped in one byte.

The envelope registers are positioned in adjacent locations. There are two types of envelope controls, direct acting controls and buffered controls. The direct acting controls always take immediate effect, and are:

- Envelope enable (reset)
- Envelope resolution (16/8 level)

The buffered controls are acted upon only at the times shown in Fig. 5 and control selection of:

- Envelope clock source
- Waveform type
- Inverted/non-inverted 'right' component

Table 1 External memory map

select	data bus inputs								operations
A0	D7	D6	D5	D4	D3	D2	D1	D0	
0	D7	D6	D5	D4	D3	D2	D1	D0	data for internal registers
1	X	X	X	A4	A3	A2	A1	A0	internal register address

Where X = don't care state.

DEVELOPMENT DATA

Table 2 Internal register map.

register address	data bus inputs D7	D6	D5	D4	D3	D2	D1	D0	operations
00	AR03	AR02	AR01	AR00	AL03	AL02	AL01	AL00	amplitude 0 right channel; left channel
01	1	1	1	1	1	1	1	1	amplitude 1 right/left
02	2	2	2	2	2	2	2	2	amplitude 2 right/left
03	3	3	3	3	3	3	3	3	amplitude 3 right/left
04	4	4	4	4	4	4	4	4	amplitude 4 right/left
05	5	5	5	5	5	5	5	5	amplitude 5 right/left
06	X	X	X	X	X	X	X	X	
07	X	X	X	X	X	X	X	X	
08	F07	F06	F05	F04	F03	F02	F01	F00	frequency of tone 0
09	1	1	1	1	1	1	1	1	frequency of tone 1
0A	2	2	2	2	2	2	2	2	frequency of tone 2
0B	3	3	3	3	3	3	3	3	frequency of tone 3
0C	4	4	4	4	4	4	4	4	frequency of tone 4
0D	F57	F56	F55	F54	F53	F52	F51	F50	frequency of tone 5
0E	X	X	X	X	X	X	X	X	
0F	X	X	X	X	X	X	X	X	
10	X	012	011	010	X	002	001	000	octave 1; octave 0
11	X	032	031	030	X	022	021	020	octave 3; octave 2
12	X	052	051	050	X	042	041	040	octave 5; octave 4
13	X	X	X	X	X	X	X	X	
14	X	X	FE5	FE4	FE3	FE2	FE1	FE0	frequency enable
15	X	X	NE5	NE4	NE3	NE2	NE1	NE0	noise enable
16	X	X	N11	N10	X	X	N01	N00	noise generator 1; noise generator 0
17	X	X	X	X	X	X	X	X	
18	E07	X	E05	E04	E03	E02	E01	E00	envelope generator 0
19	E17	X	E15	E14	E13	E12	E11	E10	envelope generator 1
1A	X	X	X	X	X	X	X	X	
1B	X	X	X	X	X	X	X	X	
1C	X	X	X	X	X	X	RST	SE	frequency reset (all channels) / sound enable (all channels)
1D	X	X	X	X	X	X	X	X	
1E	X	X	X	X	X	X	X	X	
1F	X	X	X	X	X	X	X	X	

Where:
All don't cares (X) should be written as zero's.
00 to 1F block of registers repeats eight times in the block between addresses 00 to FF (full internal memory map).

APPLICATION INFORMATION (continued)

Table 3 Register description

bit	description
ARn3; ARn2; ARn1; ARn0 (n = 0,5)	4 bits for amplitude control of right channel 0 0 0 0 minimum amplitude (off) 1 1 1 1 maximum amplitude
ALn3; ALn2; ALn1; ALn0 (n = 0,5)	4 bits for amplitude control of left channel 0 0 0 0 minimum amplitude (off) 1 1 1 1 maximum amplitude
Fn7 to Fn0 (n = 0,5)	8 bits for frequency control of the six frequency generators 0 0 0 0 0 0 0 0 lowest frequency 1 1 1 1 1 1 1 1 highest frequency
On2; On1; On0 (n = 0,5)	3 bits for octave control 0 0 0 lowest octave (31 Hz to 61 Hz) 0 0 1 (61 Hz to 122 Hz) 0 1 0 (122 Hz to 244 Hz) 0 1 1 (245 Hz to 488 Hz) 1 0 0 (489 Hz to 977 Hz) 1 0 1 (978 Hz to 1,95 kHz) 1 1 0 (1,96 kHz to 3,91 kHz) 1 1 1 highest octave (3,91 kHz to 7,81 kHz)
FEn (n = 0,5)	frequency enable bit (one tone per generator) FEn = 0 indicates that frequency 'n' is off
NEn (n = 0,5)	noise enable bit (one tone per generator) NEn = 0 indicates that noise 'n' is off
Nn1; Nn0 (n = 0,1)	2 bits for noise generator control. These bits select the noise generator rate (noise 'colour') Nn1 Nn0 clock frequency 0 0 31,3 kHz 0 1 15,6 kHz 1 0 7,6 kHz 1 1 61 Hz to 15,6 kHz (frequency generator 0/3)

DEVELOPMENT DATA

bit	description
En7; En5 to En0 (n = 0,1)	7 bits for envelope control En0 0 left and right component have the same envelope 1 right component has inverse of envelope that is applied to left component En3 En2 En1 0 0 0 zero amplitude 0 0 1 maximum amplitude 0 1 0 single decay 0 1 1 repetitive decay 1 0 0 single triangular 1 0 1 repetitive triangular 1 1 0 single attack 1 1 1 repetitive attack En4 0 4 bits for envelope control (maximum frequency = 977 Hz) 1 3 bits for envelope control (maximum frequency = 1,95 kHz) En5 0 internal envelope clock (frequency generator 1 or 4) 1 external envelope clock (address write pulse) En7 0 reset (no envelope control) 1 envelope control enabled
SE	SE sound enable for all channels (reset on power-up to 0) 0 all channels disabled 1 all channels enabled
RST	Reset signal to all frequency generators 0 all generators enabled 1 all generators reset and synchronized

Note

All rates given are based on the input of a 8 MHz clock.

APPLICATION INFORMATION (continued)

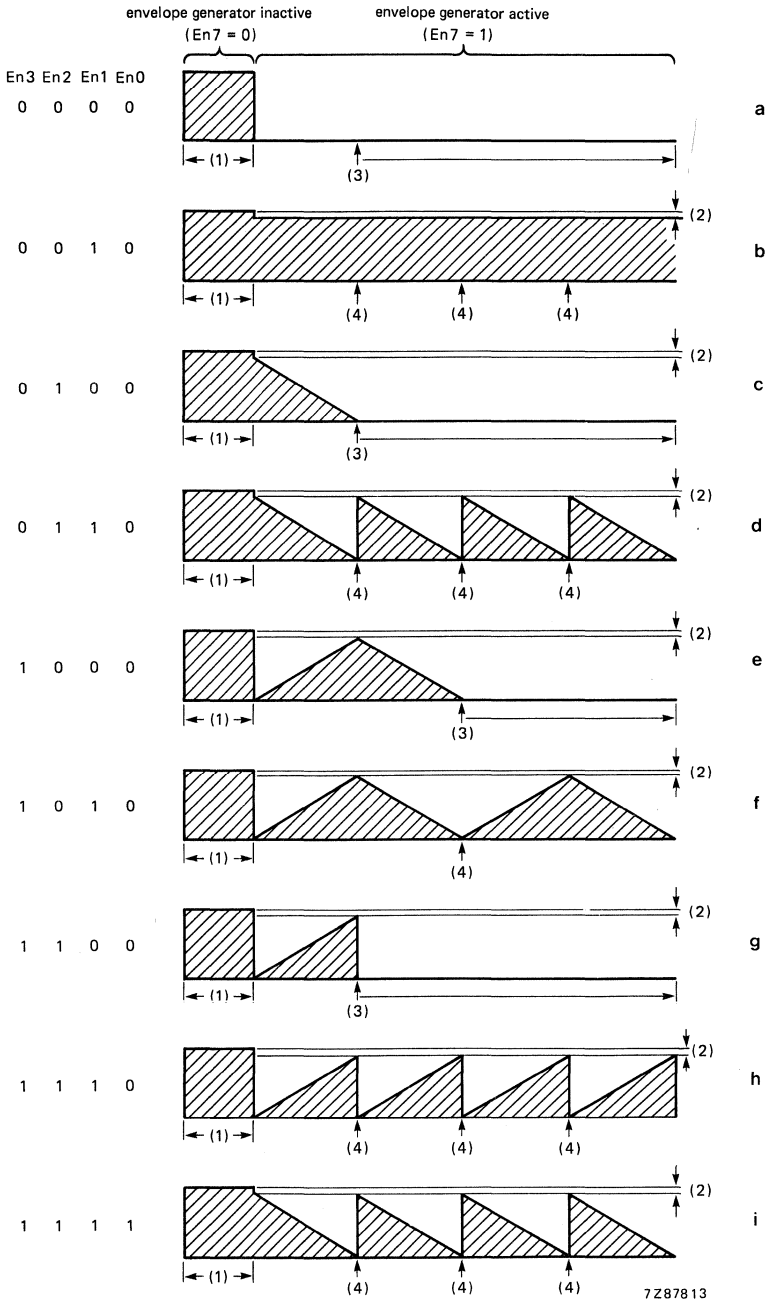


Fig. 5 Envelope waveforms.

DEVELOPMENT DATA

Notes to Fig. 5

(1) The level at this time is under amplitude control only (En7 = 0; no envelope).

(2) When the generator is active (En7 = 1) the maximum level possible is 7/8ths of the amplitude level.

(3) After position (3) the buffered controls will be acted upon when loaded.

(4) At positions (4) the buffered controls will be acted upon if already loaded.

(5) Waveforms 'a' to 'h' show the left channel (En0 = 0; left and right components have the same envelope).
Waveform 'i' shows the right channel (En0 = 1; right component inverse of envelope applied to left).

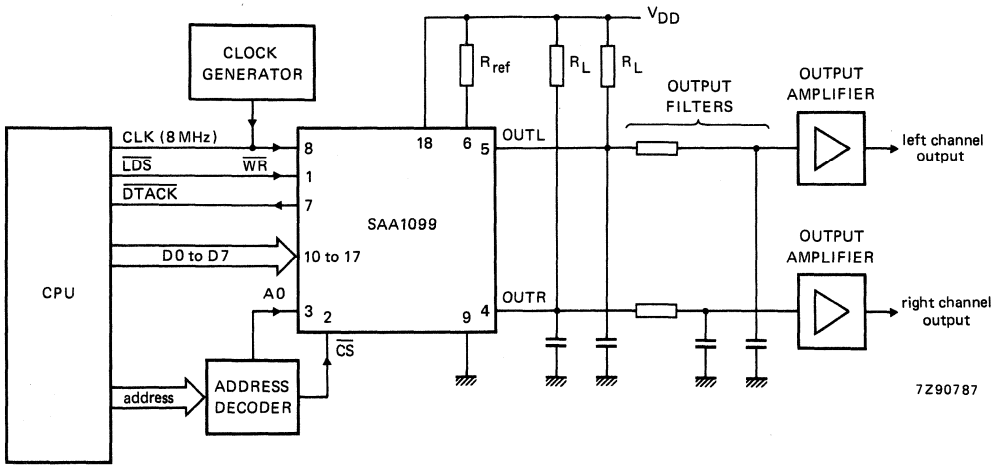


Fig. 6 Typical application circuit diagram.

MICROPROCESSOR

Originally published by Signetics April 1983

DESCRIPTION

The Signetics SCN2650A devices are 8-bit general purpose microprocessors constructed using Signetics N-channel silicon gate MOS technology. The SCN2650 series executes a fixed instruction set, with each instruction being one to three bytes in length.

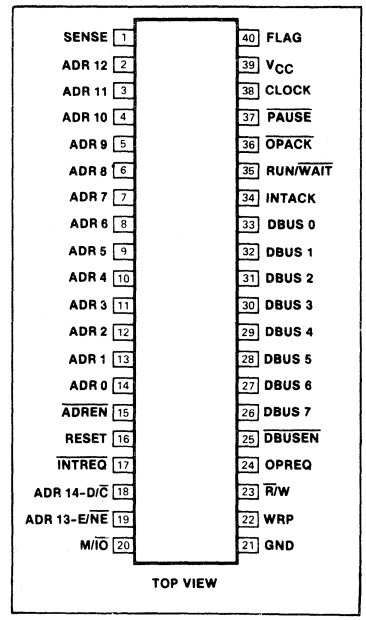
The SCN2650A contains a total of seven general purpose registers which may be used as a source or destination for arithmetic operations, as index registers, and for I/O memory transfers. An 8-level subroutine return address stack is included on the chip.

Addressing range of these processors is 32K bytes of memory and 258 I/O devices. A single level hardware vectored interrupt capability is provided.

FEATURES

- **Static 8 bit parallel NMOS microprocessor**
- **Single power supply of +5 volts**
- **TTL level single phase clock**
- **TLL compatible inputs and outputs**
- **Variable length instructions of 1, 2 or 3 bytes**
- **32K byte addressing range**
- **Coding efficiency with multiple addressing modes**
- **Synchronous or asynchronous memory and I/O interface**
- **Interfaces directly with industry standard memories**
- **Single bit serial I/O path**
- **Seven 8 bit addressable general purpose registers**
- **Vectored interrupt**
- **Subroutine return address stack**

PIN CONFIGURATION



MICROPROCESSOR BLOCK DIAGRAM

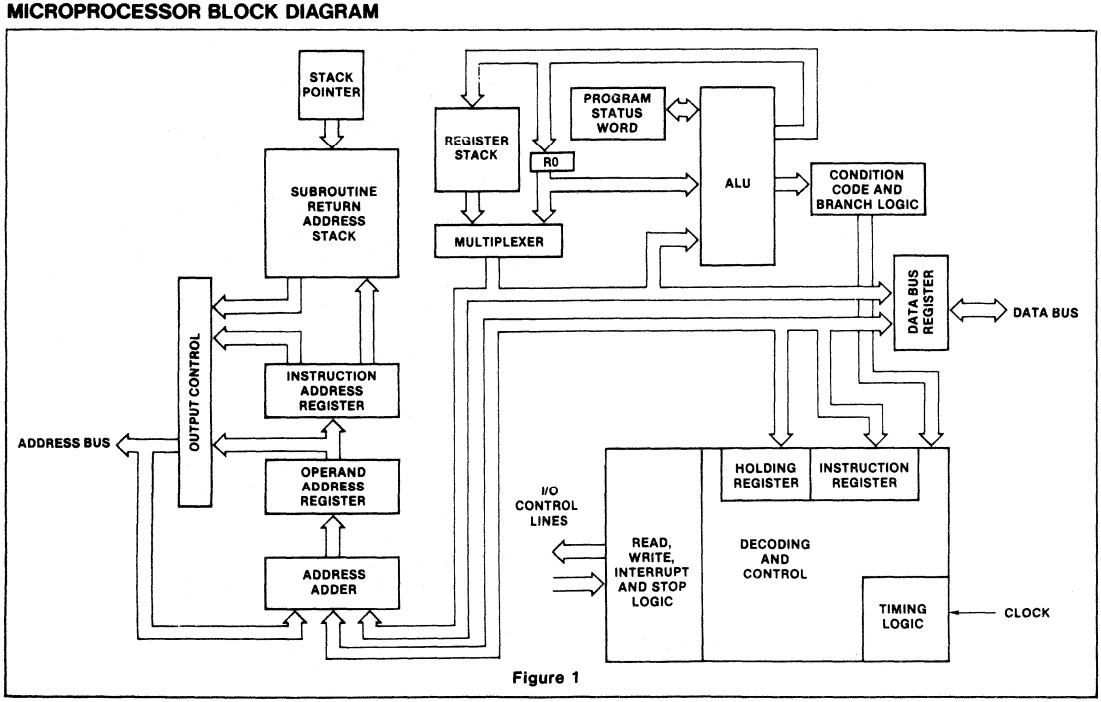


Figure 1

PIN DESIGNATION

MNEMONIC	NUMBER	NAME	TYPE	FUNCTION
ADR0-ADR12	14-2	Address lines	O	Low order memory address lines for instruction or operand fetch. ADR0 is the least significant bit and ADR12 is the most significant bit. ADR0 through ADR7 are also used as the I/O device address for extended I/O instructions.
ADR13-E/$\overline{NE}$	19	Address 13-Extended/Non extended	O	Low order memory page address line during memory reference instructions. For I/O instructions this line discriminates between extended and non-extended I/O instructions.
ADR14-D/$\overline{C}$	18	Address 14-Data/Control	O	High order memory page address line during memory reference instructions. It also serves as the I/O device address for non-extended I/O instructions.
$\overline{ADREN}$	15	Address enable	I	Active low input allowing 3-state control of the address bus ADR0-ADR12.
DBUS0-DBUS7	33-26	Data bus	I/O	These lines provide communication between the CPU, Memory, and I/O devices for instruction and data transfers.
$\overline{DBUSEN}$	25	Data bus enable	I	This active low input allows tri-state control of the data bus.
OPREQ	24	Operation request	O	Indicates to external devices that all address, data and control information is valid.
$\overline{OPACK}$	36	Operation acknowledge	I	Active low input indicating completion of an external operation. This allows asynchronous functioning of external devices.
M/$\overline{IO}$	20	Memory/input-output	O	Indicates whether the current operation references memory or I/O.
$\overline{R}$/W	23	Read/Write	O	Indicates a read or a write operation.
WRP	22	Write pulse	O	This is a timing signal from the SCN2650 that provides a positive-going pulse during each requested write operation (memory or I/O) and a high level during read operations.
SENSE	1	Sense	I	The sense bit in the PSU reflects the logic state of the sense input to the processor at pin #1.
FLAG	40	Flag	O	The flag bit in the PSU is tied to a latch that drives the flag output at pin #40.
$\overline{INTREQ}$	17	Interrupt request	I	This active low input line indicates to the processor that an external device is requesting service. The processor will recognize this signal at the end of the current instruction if the interrupt inhibit status bit is zero.
INTACK	34	Interrupt acknowledge	O	This line indicates that the SCN2650 is ready to receive the interrupt vector (relative address byte) from the interrupting device.
$\overline{PAUSE}$	37	Pause	I	This active low input is used to suspend processor operation at the end of the current instruction.
RUN/$\overline{WAIT}$	35	Run/Wait	O	This output is a processor status indicator. During normal operation this line is high. If the processor is halted either by executing a halt instruction or by a low input on the pause line, the run/wait line will go low.
RESET	16	Reset	I	Resets the instruction address register to zero. Clears interrupt inhibit.
CLOCK	38	Clock	I	A positive going pulse train that determines the instruction execution time.
V_{CC}	39	+5V supply	I	+5V power
GND	21	Ground	I	Ground

FUNCTIONAL DESCRIPTION

The SCN2650 series processors are general purpose, single chip, fixed instruction set, parallel 8-bit binary processors. A general purpose processor can perform any data manipulations through execution of a stored sequence of machine instructions. The processor has been designed to closely resemble conventional binary computers, but executes variable length instructions of one to three bytes in length.

The SCN2650 series contains a total of seven general purpose registers, each eight bits long. They may be used as source or destination for arithmetic operations, as index registers, and for I/O transfers.

The processor can address up to 32,768 bytes of memory in four pages of 8,192 bytes each. The processor instructions are one, two, or three bytes long, depending on the instruction. Variable length instructions tend to conserve memory space since a one-or-two byte instruction may often be used rather than a three byte instruction. The first byte of each instruction always specifies the operation to be performed and the addressing mode to be used. Most instructions use six of the first eight bits for this purpose, with the remaining two bits forming the register field. Some instructions use the full eight bits as an operation code.

The data bus and address signals are tristate to provide convenience in system design. Memory and I/O interface signals are asynchronous so that direct memory access (DMA) and multiprocessor operations are easy to implement.

The block diagram for the SCN2650 series (figure 1) shows the major internal components and the data paths that interconnect them. In order for the processor to execute an instruction, it performs the following general steps:

1. The instruction address register provides an address for memory.
2. The first byte of an instruction is fetched from memory and stored in the instruction register.
3. The instruction register (IR) is decoded to determine the type of instruction and the addressing mode.
4. If an operand from memory is required, the operand address is resolved and loaded into the operand address register.
5. The operand is fetched from memory and the operation is executed.
6. The first byte of the next instruction is fetched.

The instruction register holds the first byte of each instruction and directs the subsequent operations required to execute each instruction. The IR contents are decoded and used in conjunction with the timing information to control the activation and sequencing of all the other elements on the chip. The holding register is used in some multiple-byte instructions to contain further instruction information and partial absolute addresses.

The arithmetic logic unit (ALU) is used to perform all of the data manipulation operations, including load, store, add, subtract, AND, inclusive OR, exclusive OR, compare, rotate, increment and decrement. It contains and controls the carry bit, the overflow bit, the interdigit carry and the condition code register.

The register stack contains six registers that are organized into two banks of three registers each. The register select bit picks one of the two banks to be accessed by instructions. In order to accommodate the register-to register instructions, register zero (R0) is outside the array. Thus, register zero is always available along with one set of three registers.

The address adder is used to increment the instruction address and to calculate relative and indexed addresses.

The instruction address register holds the address of the next instruction byte to be accessed. The operand address register stores operand addresses and sometimes contains intermediate results during effective address calculations.

The return address stack (RAS) is a last in, first out (LIFO) storage which receives the return address whenever a branch-to-subroutine instruction is executed. When a return instruction is executed, the RAS provides the last return address for the processor's IAR. The stack contains eight levels of storage so that subroutines may be nested up to eight levels deep. The stack pointer is a three bit wraparound counter that indicates the next available level in the stack. It always points to the current address.

PROGRAM STATUS WORD

The program status word (PSW) is a major feature of the SCN2650 which greatly increases its flexibility and processing power. The PSW is a special purpose register within the processor that contains status and control bits.

It is divided into two bytes called the program status upper (PSU) and program status lower (PSL). The PSW bits may be tested, loaded, stored, preset, or cleared using the instructions which affect the PSW. The bits are utilized as shown in table 1.

Table 1 PROGRAM STATUS WORD

PSU0,1,2	SP	Pointer for the return address stack.
PSU3,4		Not used. These bits are always zero.
PSU5	II	Used to inhibit recognition of additional Interrupts.
PSU6	F	Flag is a latch directly driving the flag output.
PSU7	S	Sense equals the state of the sense input.
PSL0	C	Carry stores any carry from the high-order bit of ALU.
PSL1	COM	Compare determines if a logical or arithmetic comparison is to be made.
PSL2	OVF	Overflow is set if a two's complement overflow occurs.
PSL3	WC	With carry determines if the carry is used in arithmetic and rotate instructions.
PSL4	RS	Register select identifies which bank of 3.GP registers is being used.
PSL5	IDC	Inter digit carry stores the bit-3 to bit-4 carry in arithmetic operations.
PSL6,7	CC	Condition code is affected by compare, test and arithmetic instructions.

PSU

7	6	5	4	3	2	1	0
S	F	II	—	—	SP2	SP1	SP0

S Sense
F Flag
II Interrupt inhibit
SP2 Stack pointer two
SP1 Stack pointer one
SP0 Stack pointer zero

PSL

7	6	5	4	3	2	1	0
CC1	CC0	IDC	RS	WC	OVF	COM	C

CC1 Condition code one
CC0 Condition code zero
IDC Interdigit carry
RS Register bank select
WC With/without carry
OVF Overflow
COM Logical arithmetic compare
C Carry/borrow

INPUT/OUTPUT INTERFACE

The SCN2650 series microprocessor has a set of versatile I/O instructions and can perform I/O operations in a variety of ways. One and two byte I/O instructions are provided, as well as a special single-bit I/O facility. The I/O modes provided by the SCN2650 are designated as data, control, and extended I/O.

Data or control I/O instructions, also called non-extended I/O instructions, are one byte long. Any general purpose register can be used as the source or destination. A special control line indicates if either a data or control instruction is being executed.

Extended I/O is a two-byte read or write instruction. Execution of an extended I/O instruction will cause an 8-bit address, taken from the second byte of the instruction, to be placed on the low order eight address lines. The data, which can originate or terminate with any general purpose register, is placed on the data bus. This type of I/O can be used to simultaneously select a device and send data to it.

Memory reference instructions that address data outside of physical memory may also be used for I/O operations. When an instruction is executed, the address may be decoded by the I/O device rather than memory.

MEMORY INTERFACE

The memory interface consists of the address bus, the 8-bit data bus and several signals that operate in an interlocked or handshaking mode.

The write pulse signal is designed to be used as a memory strobe signal for any memory type. It has been particularly optimized to be used as the chip enable or read/write signal.

INTERRUPT HANDLING CAPABILITY

The SCN2650 series has a single level hardware vectored interrupt capability. When an interrupt occurs, the processor finishes the current instruction and sets the interrupt inhibit bit in the PSW. The processor then executes a branch to subroutine relative to location zero (ZBSR) instruction and sends out interrupt acknowledge and operation request signals. On receipt of the INTACK signal, the interrupt device inputs an 8-bit address, the interrupt vector, on the data bus. The relative and relative indirect addressing modes combined with this 8-bit address allow interrupt service routines to begin at any addressable memory location.

INSTRUCTION SET

The 2650 instruction set consists of many powerful instructions which are all easily understood and are typical of larger computers. There are one-, two-, and three-byte instructions as a result of the multiplicity of addressing modes.

Automatic incrementing or decrementing of an index register is available in the arithmetic indexed instructions. All of the branch instructions except indexed branching can be conditional.

Register-to-register instructions are one byte; register-to-storage instructions are two or three bytes long. The two-byte register-to-memory instructions are either immediate or relative addressing types.

FOR DETAILED INFORMATION SEE RELEVANT DATA BOOK OR DATA SHEET

PAL COLOUR ENCODER AND VIDEO SUMMER

The TEA1002 is mainly intended for video games, add-on teletext applications and colour bar generators for video test equipment. It is a bipolar integrated circuit which converts binary colour information into a PAL composite video output suitable for driving a v.h.f./u.h.f. modulator.

QUICK REFERENCE DATA

Supply voltage (pin 10)	$V_P = V_{10\text{-}16}$	nom.	12 V
Supply current at V_P = 12 V	$I_P = I_{10}$	typ.	70 mA
Input voltages (pins, 1, 2, 3, 4, 5, 12, 15, 18)			
LOW	V_{IL}	$\leqslant$	0,8 V
HIGH	V_{IH}	$\geqslant$	2,0 V
Composite video output voltage (pin 8)			
peak-to-peak value	$V_{8\text{-}16(p\text{-}p)}$	typ.	3 V
Operating ambient temperature range	T_{amb}		−20 to +65 °C

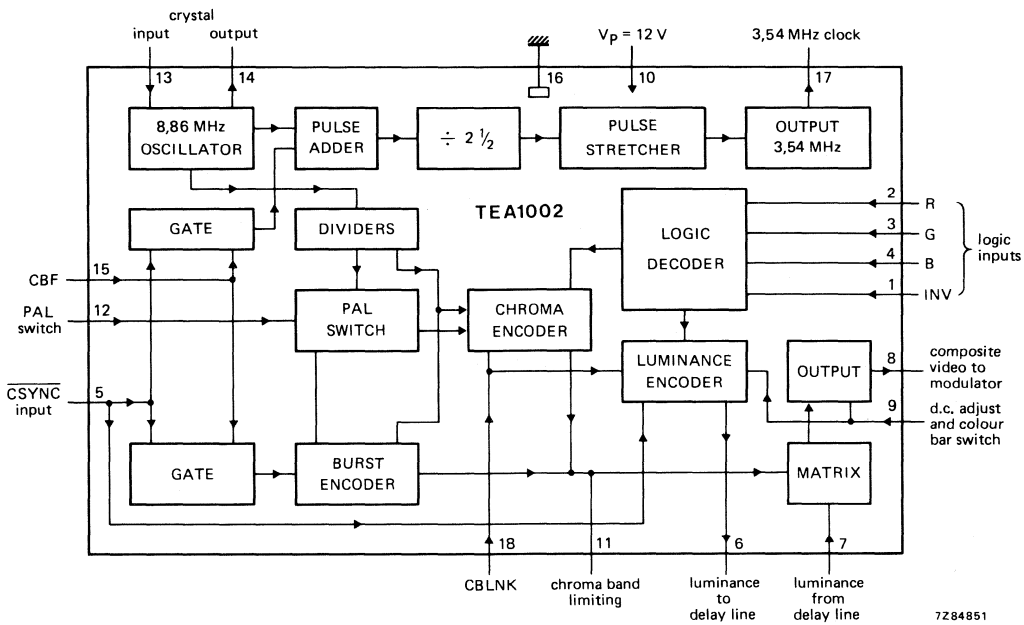


Fig. 1 Block diagram.

PACKAGE OUTLINE

18-lead DIL; plastic (SOT-102CS).

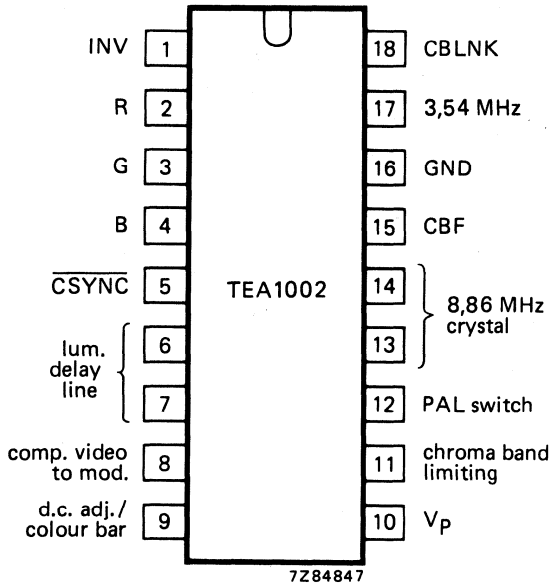


Fig. 2 Pinning diagram.

PINNING

1	INV	inverting logic input
2	R	red logic input
3	G	green logic input
4	B	blue logic input
5	$\overline{\text{CSYNC}}$	composite sync input
6	luminance output to delay line	
7	luminance input from delay line	
8	composite video output to modulator	
9	d.c. adjust and colour bar switch	
10	V_P	supply voltage (+ 12 V)
11	chroma band limiting	
12	PAL switch input	
13	crystal input } 8,86 MHz	
14	crystal output }	
15	CBF	colour burst flag input
16	GND	ground
17	3,54 MHz clock output	
18	CBLNK	composite blanking input

GENERAL DESCRIPTION

The TEA1002 PAL colour encoder and video summer IC has an internal 8,86 MHz oscillator from which the 4,43 MHz (R-Y) and B-Y) waveforms are generated. For use in TV games systems, a 3,54 MHz clock output is provided which is buffered via the 2621 sync generator IC. The TEA1002 accepts timing signals (composite sync burst gate, PAL switch and composite blanking) from the 2621 and 4-bit binary coded logic inputs giving colour information from the 2636 programmable video interface IC. The resulting output, which has an adjustable d.c. level, is a 16 colour (including black and white) composite video signal, based on 75% colour bars. Alternatively, with one of the colour inputs connected to ground and the d.c. adjustment disabled, the TEA1002 can be used as a general purpose video encoder providing standard 95% colour bars from RGB logic inputs, suitable for applications such as add-on teletext.

RATINGS

Limiting values in accordance with the Absolute Maximum System (IEC 134)

Supply voltage (pin 10)	$V_P = V_{10\text{-}16}$	max. 13,2 V
Input voltage (pins 1, 2, 3, 4, 5, 12, 15, 18) HIGH	V_{IH}	max. V_P V
Storage temperature range	T_{stg}	−25 to + 125 °C
Operating ambient temperature range	T_{amb}	−20 to + 65 °C

PACKAGE OUTLINES

GENERAL INFORMATION

INTRODUCTION

The following information applies to all packages unless otherwise specified on individual package outline drawings.

General

1. Dimensions shown are metric units (millimeters), except those in parentheses which are English units (inches).
2. Lead spacing shall be measured within this zone.
   a. Shoulder and lead tip dimensions are to centerline of leads.
3. Tolerances non-cumulative
4. Thermal resistance values are determined by utilizing the linear temperature dependence of the forward voltage drop across the substrate diode in a digital device to monitor the junction temperature rise during known power application across V_{CC} and ground. The values are based upon 120 mils square die for plastic packages and a 90 mils square die in the smallest available cavity for hermetic packages. All units were solder mounted to P.C. boards, with standard stand-off, for measurement.

Plastic Only

5. Lead material: Olin 194 (copper alloy) or equivalent, solder dipped.
6. Body material: Plastic (epoxy)
7. Round hole in top corner denotes lead No. 1.
8. Body dimensions do not include molding flash.
9. SO packages-microminiature packages:
   a. Lead material: Olin 194 (copper alloy) or equivalent, solder dipped.
   b. Body material: Plastic (epoxy).

Hermetic Only

10. Lead material
    a. ASTM alloy F-15 (KOVAR) or equivalent—gold plated, tin plated, or solder dipped.
    b. ASTM alloy F-30 (Alloy 42) or equivalent—tin plated, gold plated or solder dipped.
    c. ASTM alloy F-15 (KOVAR) or equivalent—gold plated.
11. Body Material
    a. Eyelet, ASTM alloy F-15 or equivalent—gold or tin plated, glass body.
    b. Ceramic with glass seal at leads.
    c. BeO ceramic with glass seal at leads.
    d. Ceramic with ASTM alloy F-30 or equivalent.
12. Lid Material
    a. Nickel or tin plated nickel, weld seal
    b. Ceramic, glass seal.
    c. ASTM alloy F-15 or equivalent, gold plated, alloy seal.
    d BeO Ceramic with glass seal.
13. Signetics symbol, angle cut, or lead tab denotes Lead No. 1.
14. Recommended minimum offset before lead bend.
15. Maximum glass climb .010 inches.
16. Maximum glass climb or lid skew is .010 inches.
17. Typical four places.
18. Dimension also applies to seating plane.

PLASTIC PACKAGES

NO. OF LEADS	PACKAGE CODE	Θ_{ja}/Θ_{jc} (°C/W)	DESCRIPTION
Plastic Dual-In-Line			
14	NHA	95/33	Cu Lead Frame
16	NJA	73/33	Cu Lead Frame
18	NKA	69/26	Cu Lead Frame
20	NLA	65/26	Cu Lead Frame
24	NNA	60/23	Cu Lead Frame
28	NQA	56/21	Cu Lead Frame
SO Packages			
16	DJ	tbd	SO-16L
20	DL	tbd	SO-20L

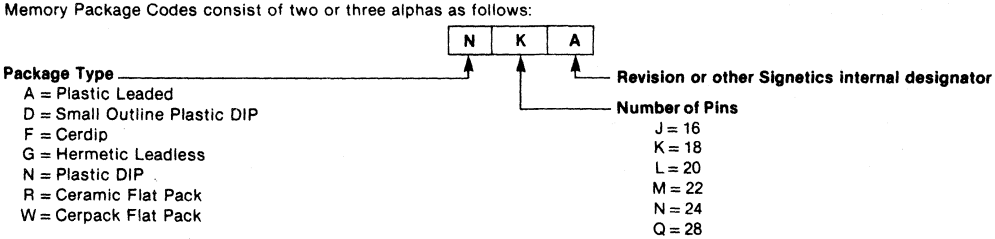
GENERAL INFORMATION (continued)

HERMETIC PACKAGES

NO. OF LEADS	PACKAGE CODE	Θ_{ja}/Θ_{jc} (°C/W)	DESCRIPTION[1]
Flat Packs			
10	QF	230/55	Flat Ceramic
10	WF	240/50	Flat Ceramic
14	QHA	185/45	Flat Ceramic Laminate
14	WH	205/50	Flat Ceramic
16	QJA	170/45	Flat Ceramic Laminate
16	RJA	133/30	Flat Ceramic, BeO
16	WJ	200/50	Flat Ceramic
18	RKA	107/22	Flat Ceramic, BeO
24	QNA	155/44	Flat Ceramic Laminate
24	RNA	107/22	Flat Ceamic, BeO
24	WN	155/40	Flat Ceramic
28	RQA	107/22	Flat Ceramic, BeO
40	RWA	95/20	Flat Ceramic, BeO
Cerdip Family			
8	FE	110/30	Dual-In-Line Ceramic
14	FH	110/30	Dual-In-Line Ceramic
16	FJ	77/30	Dual-In-Line Ceramic
18	FK	73/27	Dual-In-Line Ceramic
20	FL	72/25	Dual-In-Line Ceramic
22	FM	66/27	Dual-In-Line Ceramic
24	FN	63/26	Dual-In-Line Ceramic
28	FQ	57/27	Dual-In-Line Ceramic

NOTE
1. Dual-In-Line packages unless otherwise described.

Memory Package Codes consist of two or three alphas as follows:



PLASTIC: Leaded chip carrier

A (28-PLCC)

For current information
contact local sales offices

A (44-PLCC)

For current information
contact local sales offices

A (52-PLCC)

For current information
contact local sales offices

A (68-PLCC)

For current information
contact local sales offices

A (84-PLCC)

For current information
contact local sales offices

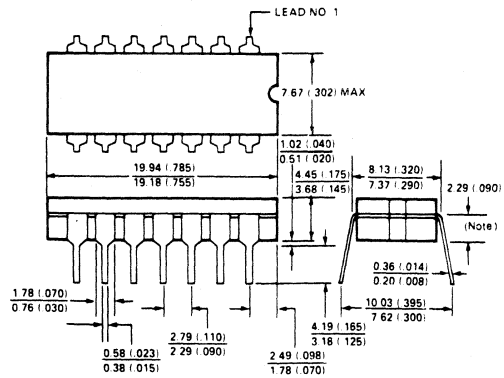
PLASTIC: SO dual-in-line

D (SO-16)

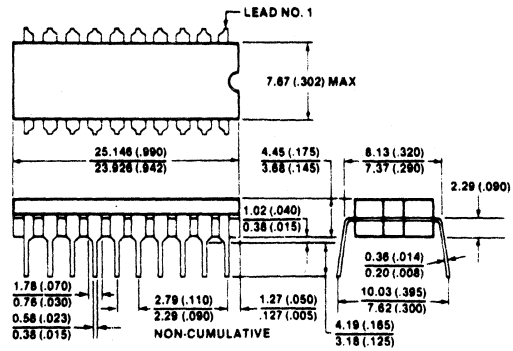
See SOT-109A.

HERMETIC: Cerdip

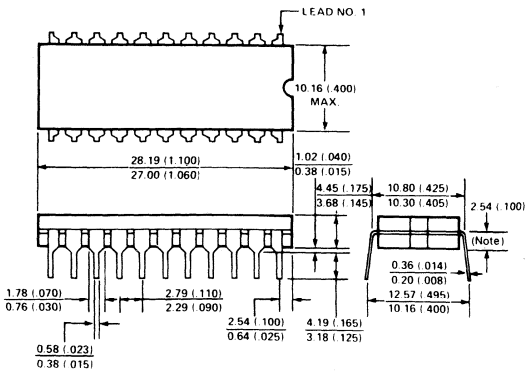
FH 14



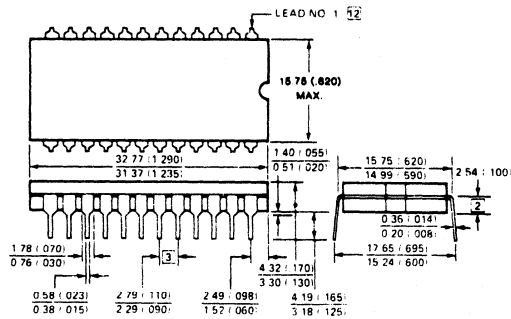
FL 20



FM 22

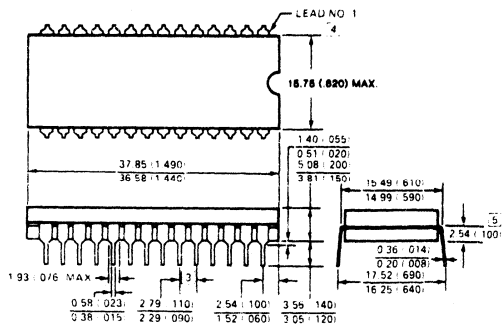


FN 24



CONSTRUCTION NOTES 9c 10d 11c

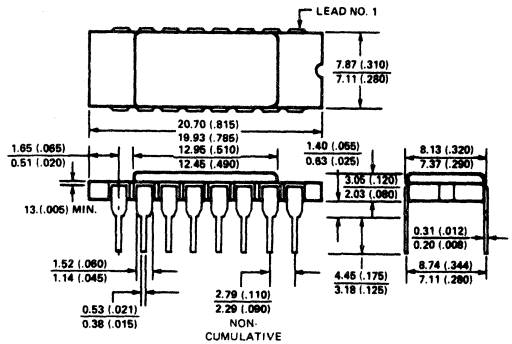
FQ 28



CONSTRUCTIONS NOTES 9c 10d 11c

HERMETIC: Cerdil

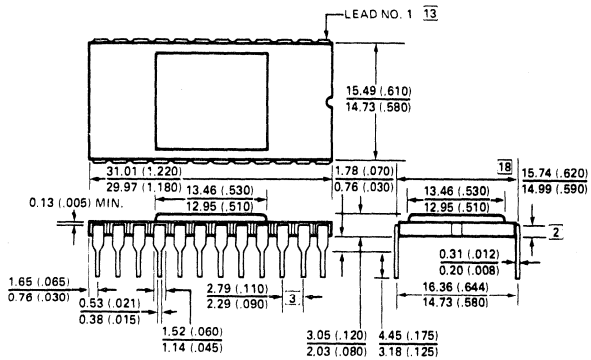
I 16



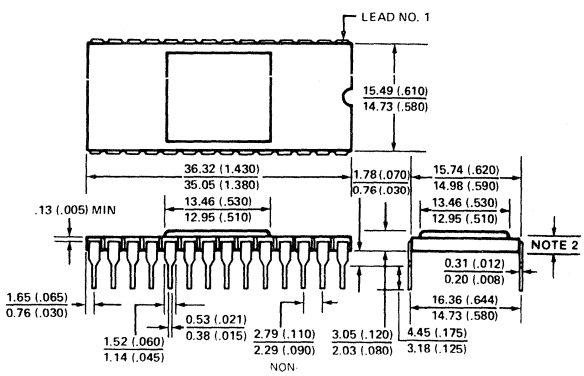
I 20

For current information contact local sales offices

I 24

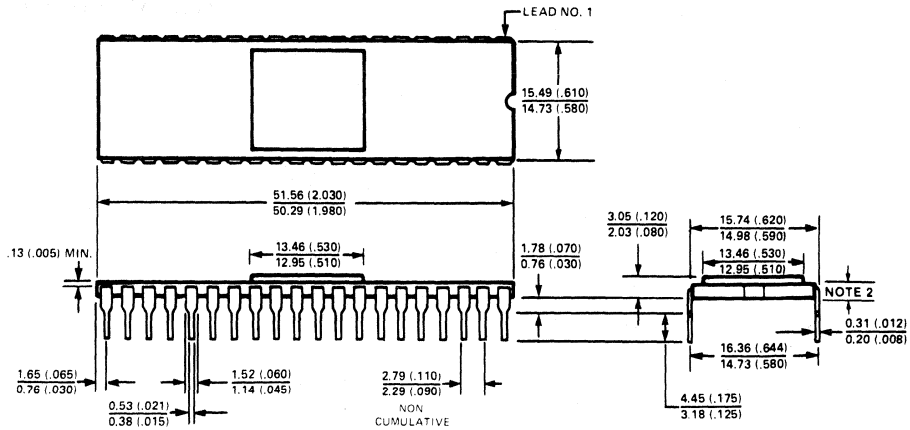


I 28

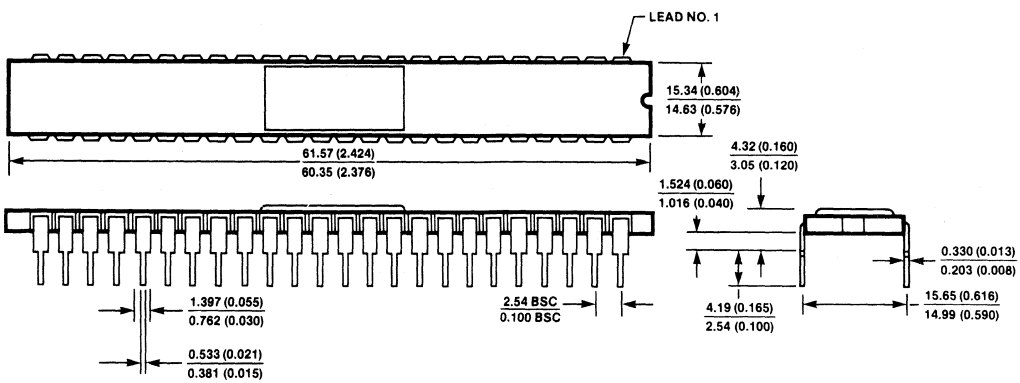


HERMETIC: Cerdil (continued)

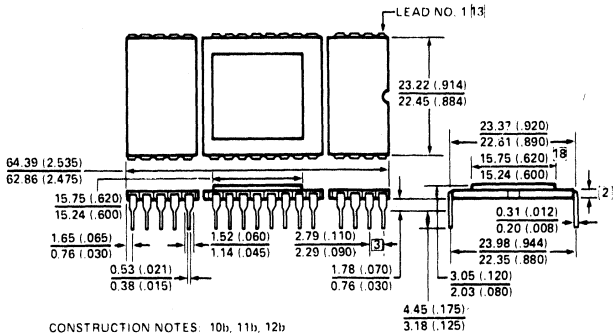
I 40



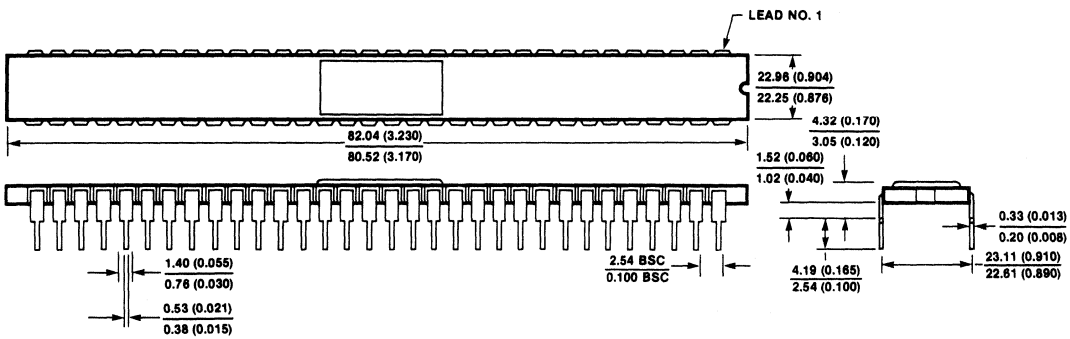
I 48



I 50



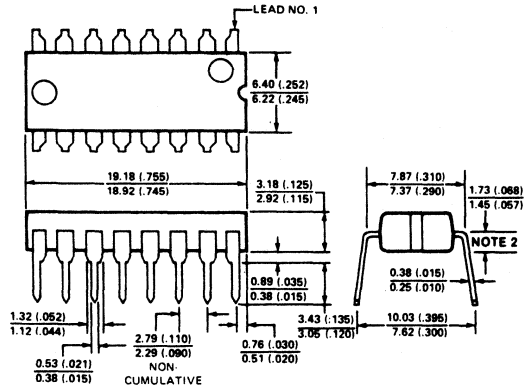
I 64



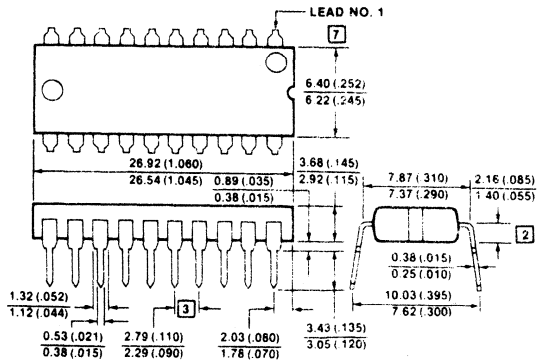
NOTE:
1. If solder dipped terminals used, terminal dimension tolerances may be increased by .001.

PLASTIC: Dual-in-line

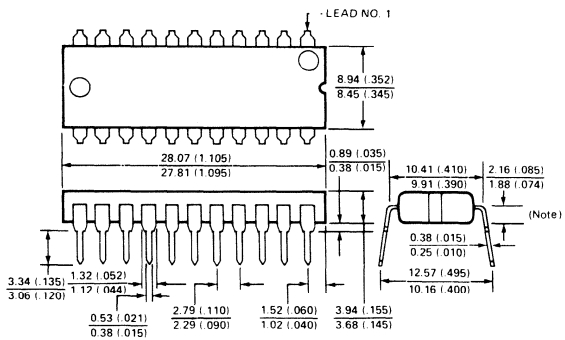
NJA 16



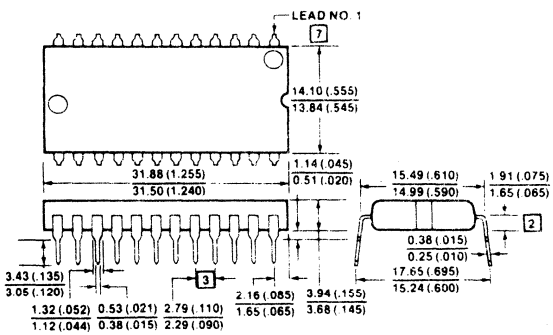
NLA 20



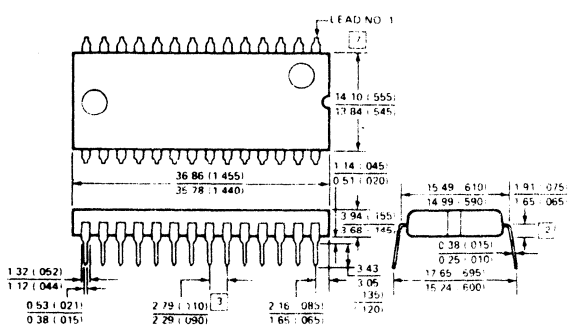
N 22



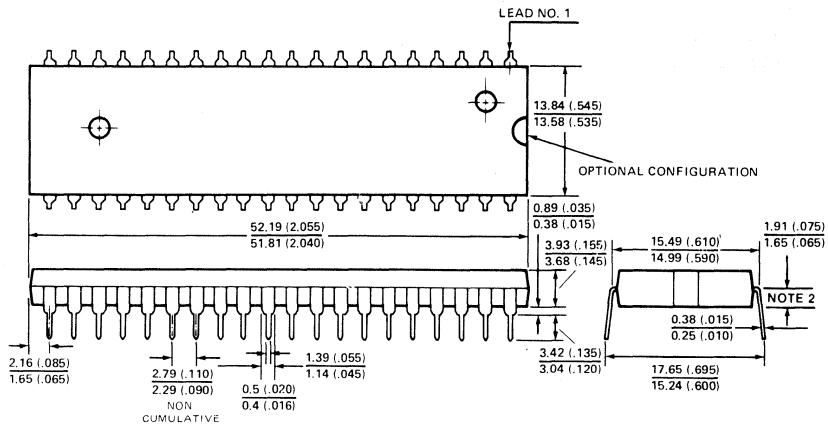
NNA 24



NQA 28



N 40



N 48

For current information
contact local sales offices

N 50

For current information
contact local sales offices

N 64

For current information
contact local sales offices

Pin Grid Array

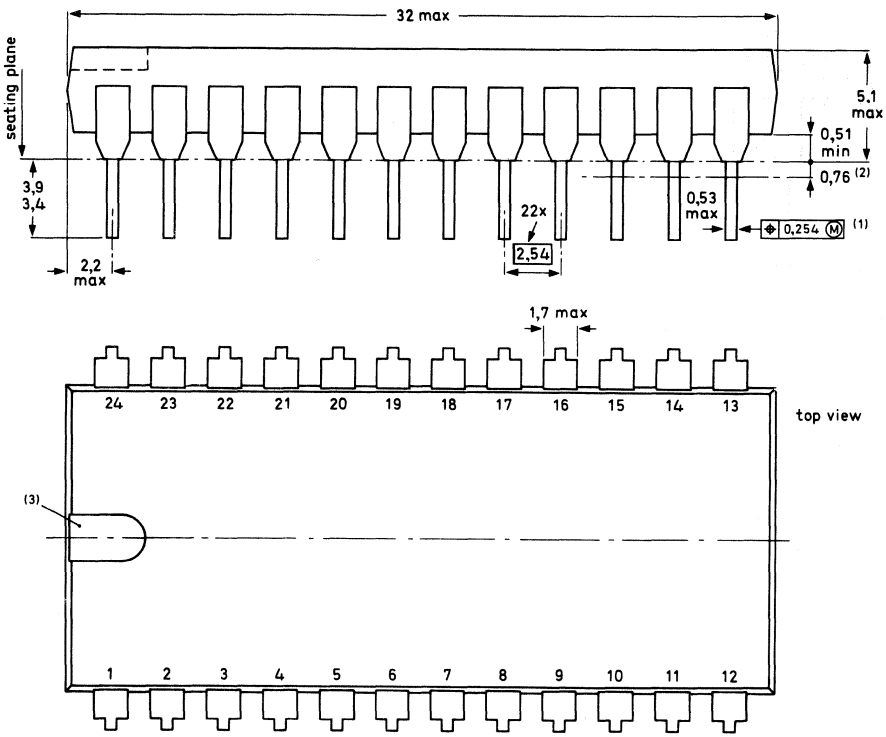
P 68 PGA

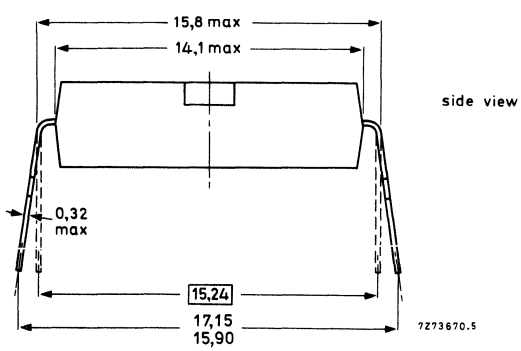
For current information
contact local sales offices

P 84 PGA

For current information
contact local sales offices

24-LEAD DUAL IN-LINE; PLASTIC (SOT-101A)





⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

(2) Lead spacing tolerances apply from seating plane to the line indicated.

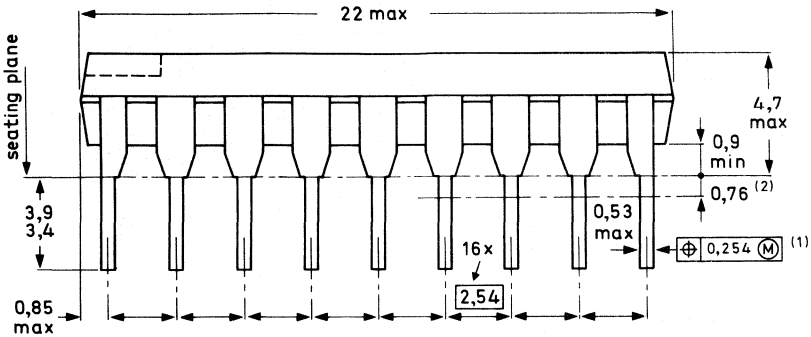
(3) Index may be horizontal as shown, or vertical.

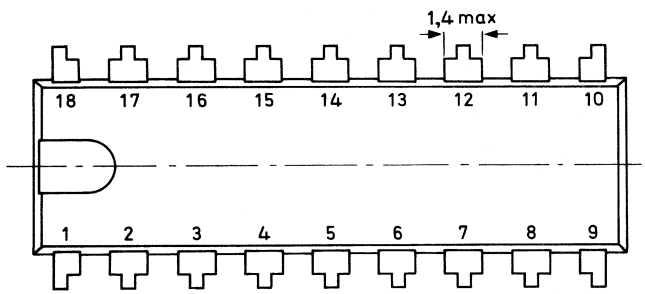
Dimensions in mm

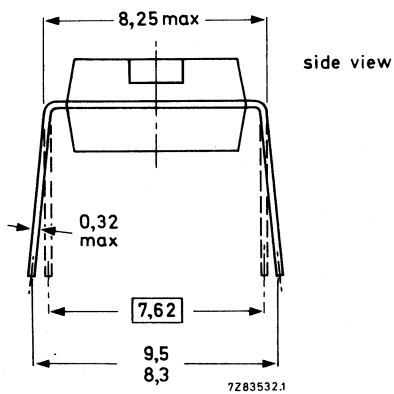
SOLDERING

See page 1344.

18-LEAD DUAL IN-LINE; PLASTIC (SOT-102CS)







⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

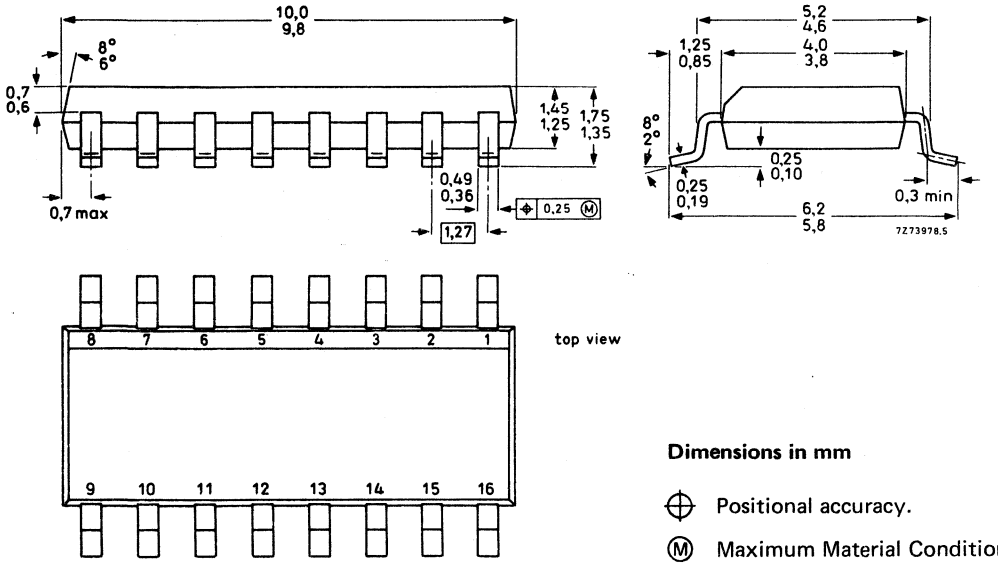
(2) Lead spacing tolerances apply from seating plane to the line indicated.

Dimensions in mm

SOLDERING

See page 1344.

16-LEAD MINI-PACK; PLASTIC (SO-16; SOT-109A)



Dimensions in mm

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

SOLDERING

The reflow solder technique

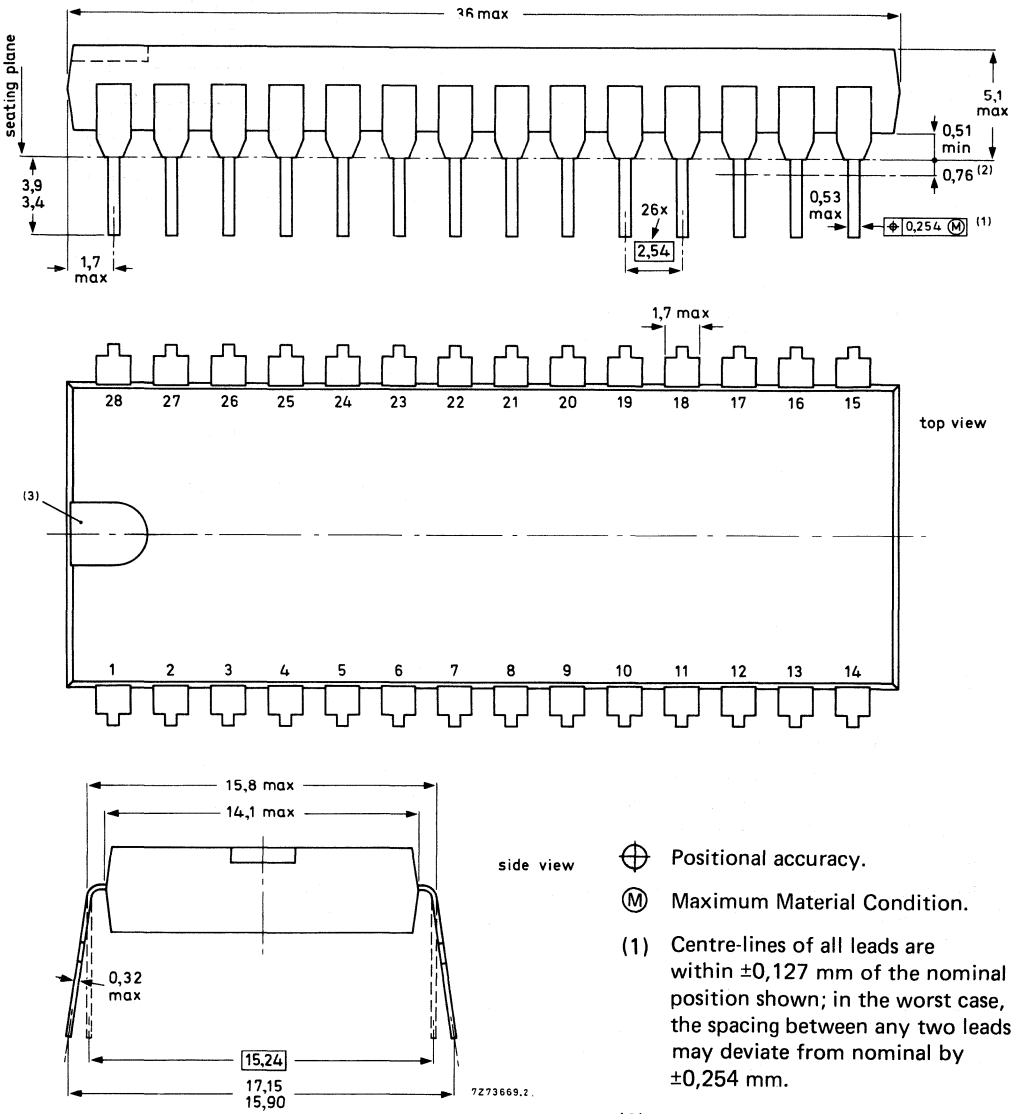
The preferred technique for mounting miniature components on hybrid thick or thin-film circuits is reflow soldering. Solder is applied to the required areas on the substrate by dipping in a solder bath or, more usually, by screen printing a solder paste. Components are put in place and the solder is reflowed by heating.

Solder pastes consist of very finely powdered solder and flux suspended in an organic liquid binder. They are available in various forms depending on the specification of the solder and the type of binder used. For hybrid circuit use, a tin-lead solder with 2 to 4% silver is recommended. The working temperature of this paste is about 220 to 230 °C when a mild flux is used.

For printing the paste onto the substrate a stainless steel screen with a mesh of 80 to 105 μm is used for which the emulsion thickness should be about 50 μm. To ensure that sufficient solder paste is applied to the substrate, the screen aperture should be slightly larger than the corresponding contact area.

The contact pins are positioned on the substrate, the slight adhesive force of the solder paste being sufficient to keep them in place. The substrate is heated to the solder working temperature preferably by means of a controlled hot plate. The soldering process should be kept as short as possible: 10 to 15 seconds is sufficient to ensure good solder joints and evaporation of the binder fluid.
After soldering, the substrate must be cleaned of any remaining flux.

28-LEAD DUAL IN-LINE; PLASTIC (SOT-117)



⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

(2) Lead spacing tolerances apply from seating plane to the line indicated.

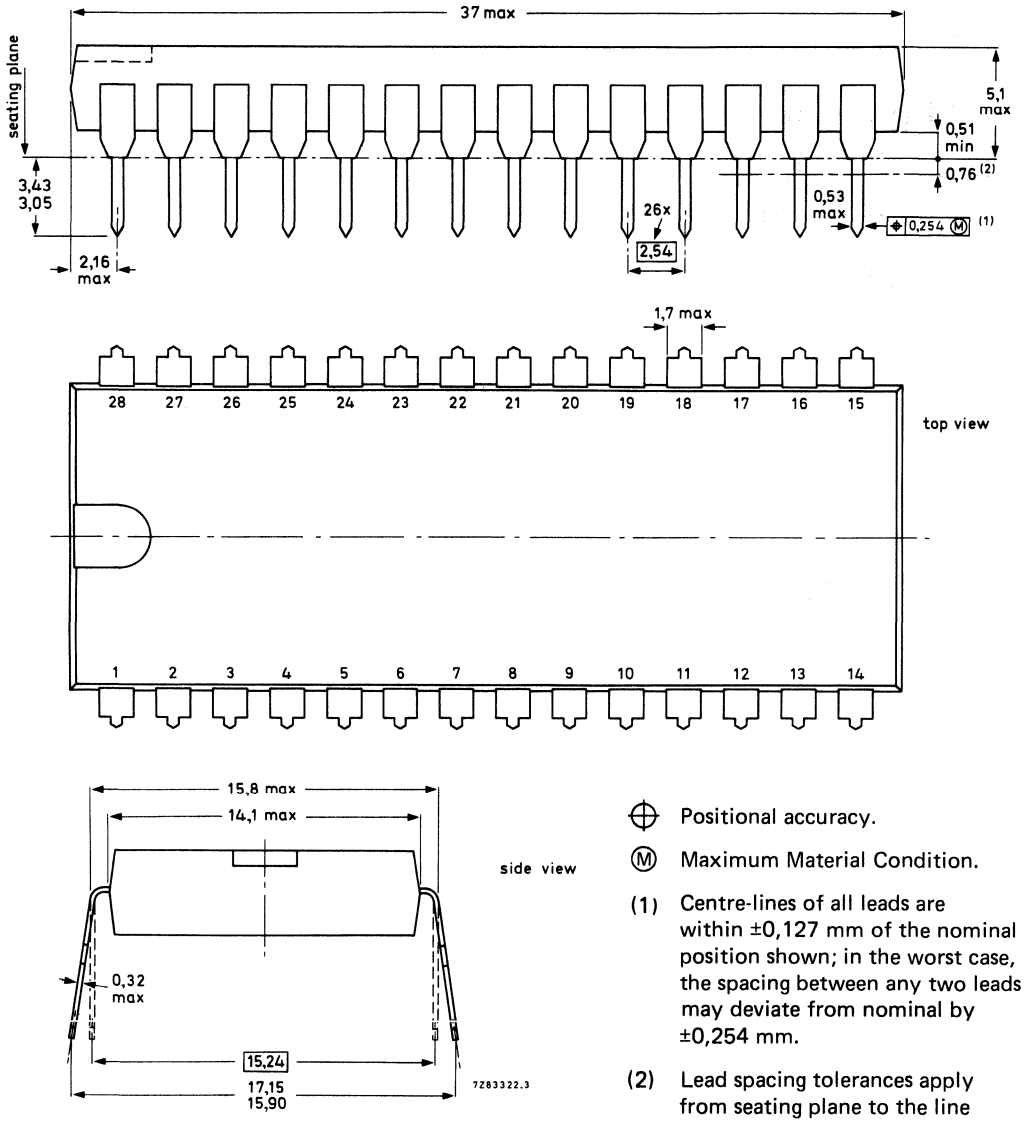
(3) Index may be horizontal as shown, or vertical.

Dimensions in mm

SOLDERING

See page 1344.

28-LEAD DUAL IN-LINE; PLASTIC (SOT-117D)



⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

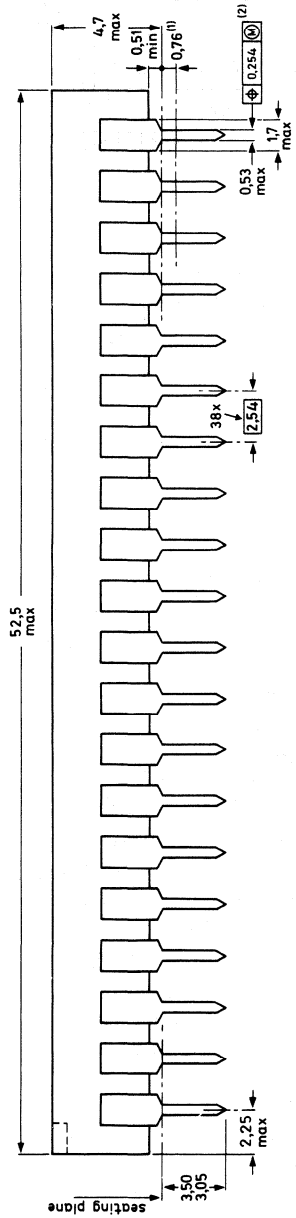
(2) Lead spacing tolerances apply from seating plane to the line indicated.

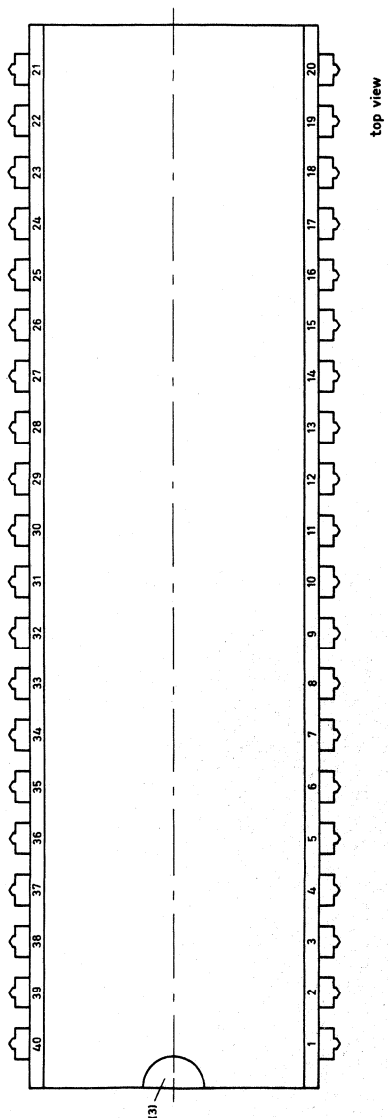
Dimensions in mm

SOLDERING

See page 1344.

40-LEAD DUAL IN-LINE; PLASTIC (SOT-129)





⊕ Positional accuracy.

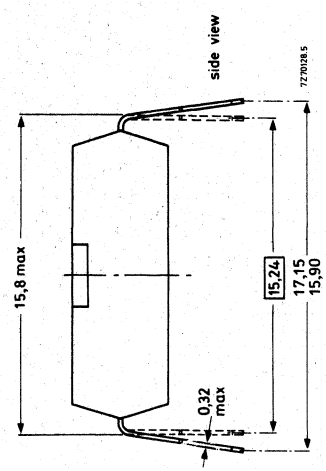
Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

(2) Lead spacing tolerances apply from seating plane to the line indicated.

(3) Index may be horizontal as shown, or vertical.

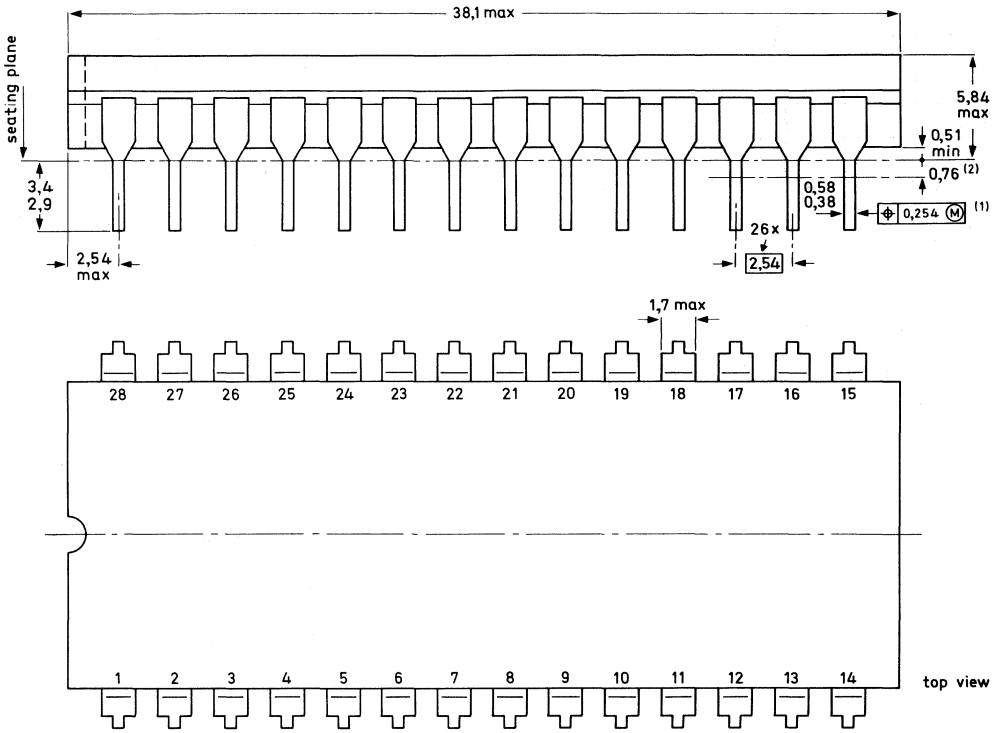
Dimensions in mm

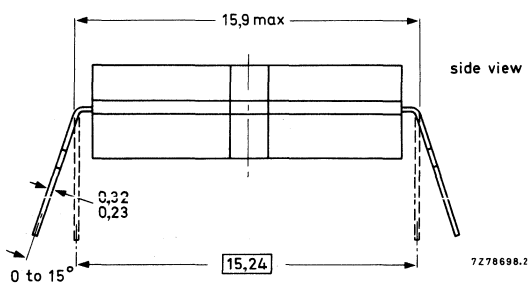


SOLDERING

See page 1344.

28-LEAD DUAL IN-LINE; CERAMIC (CERDIP) (SOT-135A)





⊕ Positional accuracy.

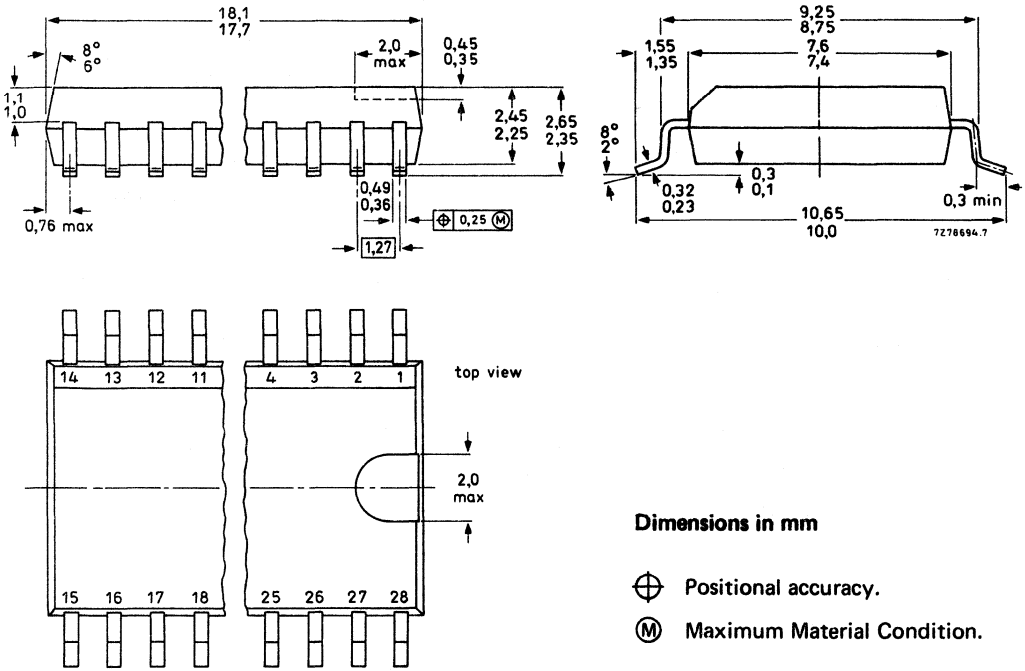
Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

(2) Lead spacing tolerances apply from seating plane to the line indicated.

Dimensions in mm

28-LEAD MINI-PACK; PLASTIC (SO-28; SOT-136A)



Dimensions in mm

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

SOLDERING

The reflow solder technique

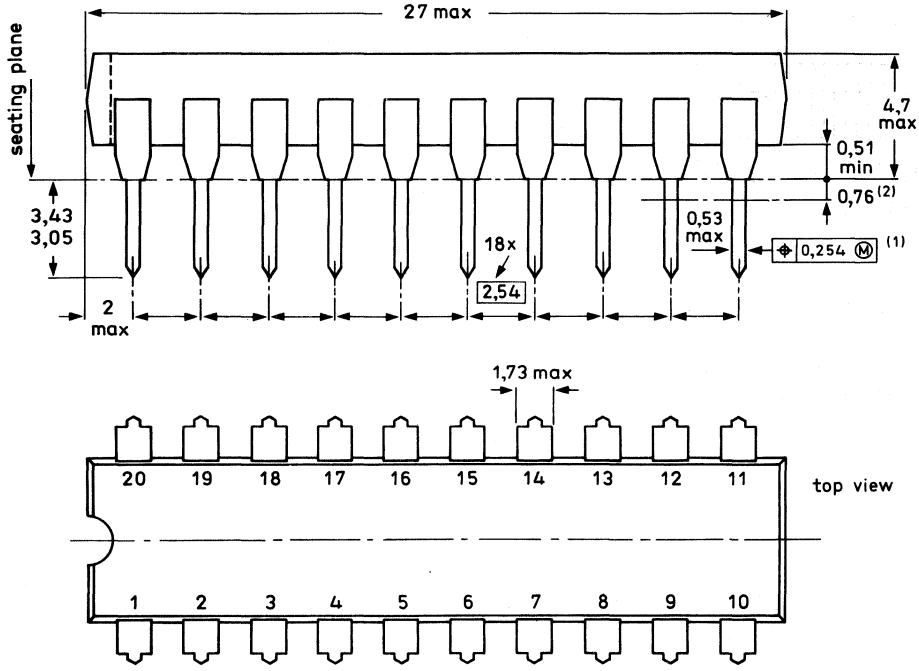
The preferred technique for mounting miniature components on hybrid thick or thin-film circuits is reflow soldering. Solder is applied to the required areas on the substrate by dipping in a solder bath or, more usually, by screen printing a solder paste. Components are put in place and the solder is reflowed by heating.

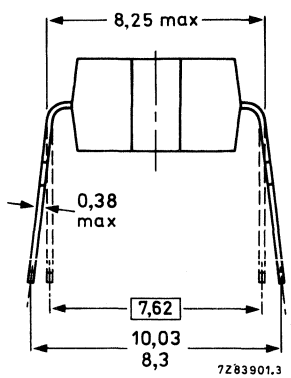
Solder pastes consist of very finely powdered solder and flux suspended in an organic liquid binder. They are available in various forms depending on the specification of the solder and the type of binder used. For hybrid circuit use, a tin-lead solder with 2 to 4% silver is recommended. The working temperature of this paste is about 220 to 230 °C when a mild flux is used.

For printing the paste onto the substrate a stainless steel screen with a mesh of 80 to 105 μm is used for which the emulsion thickness should be about 50 μm. To ensure that sufficient solder paste is applied to the substrate, the screen aperture should be slightly larger than the corresponding contact area.

The contact pins are positioned on the substrate, the slight adhesive force of the solder paste being sufficient to keep them in place. The substrate is heated to the solder working temperature preferably by means of a controlled hot plate. The soldering process should be kept as short as possible: 10 to 15 seconds is sufficient to ensure good solder joints and evaporation of the binder fluid.
After soldering, the substrate must be cleaned of any remaining flux.

20-LEAD DUAL IN-LINE; PLASTIC (SOT-146; 146C1)





side view

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

(1) Centre-lines of all leads are within ±0,127 mm of the nominal position shown; in the worst case, the spacing between any two leads may deviate from nominal by ±0,254 mm.

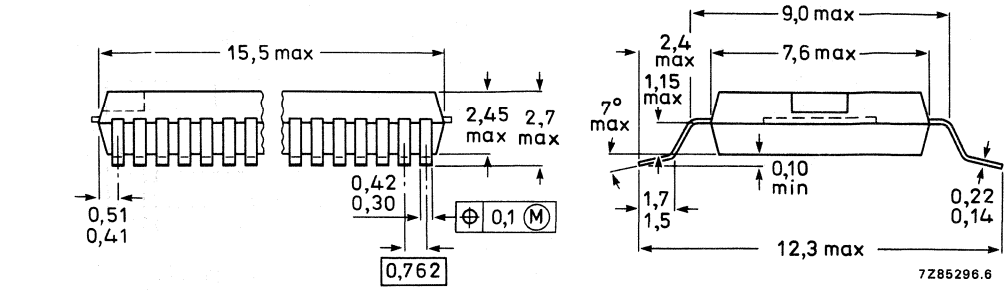
(2) Lead spacing tolerances apply from seating plane to the line indicated.

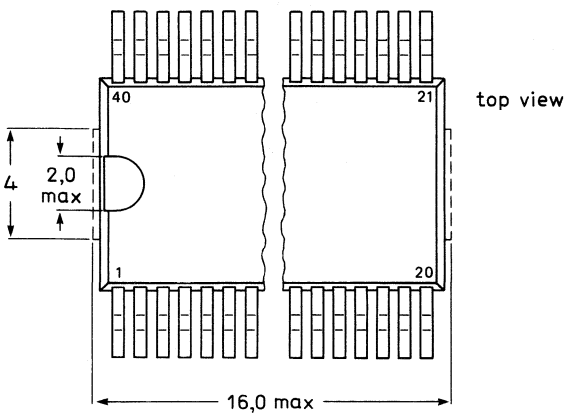
Dimensions in mm

SOLDERING

See page 1344.

40-LEAD MINI-PACK; PLASTIC (VSO-40; SOT-158A)





Dimensions in mm

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

SOLDERING

The reflow solder technique

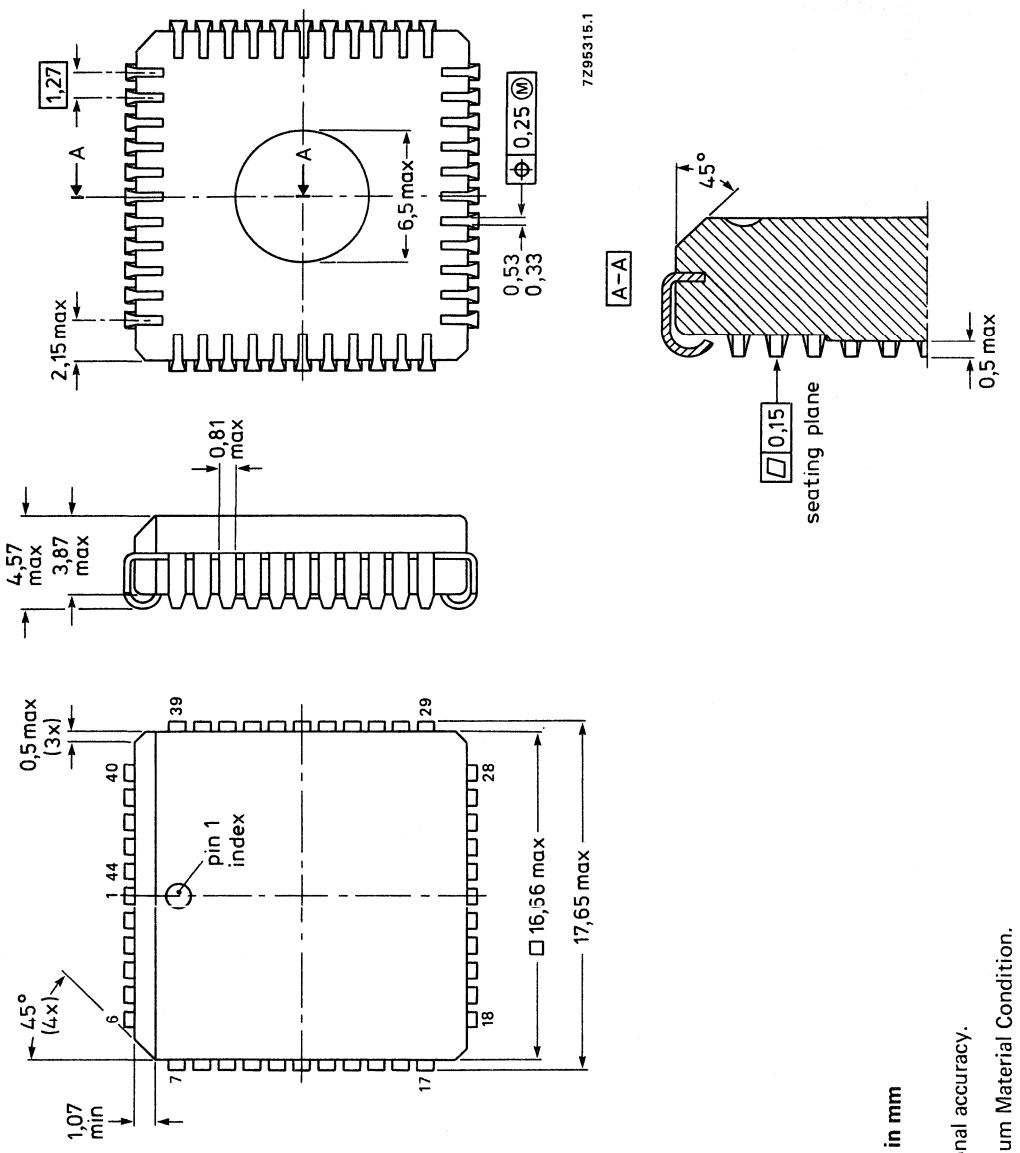
The preferred technique for mounting miniature components on hybrid thick or thin-film circuits is reflow soldering. Solder is applied to the required areas on the substrate by dipping in a solder bath or, more usually, by screen printing a solder paste. Components are put in place and the solder is reflowed by heating.

Solder pastes consist of very finely powdered solder and flux suspended in an organic liquid binder. They are available in various forms depending on the specification of the solder and the type of binder used. For hybrid circuit use, a tin-lead solder with 2 to 4% silver is recommended. The working temperature of this paste is about 220 to 230 °C when a mild flux is used.

For printing the paste onto the substrate a stainless steel screen with a mesh of 80 to 105 μm is used for which the emulsion thickness should be about 50 μm. To ensure that sufficient solder paste is applied to the substrate, the screen aperture should be slightly larger than the corresponding contact area.

The contact pins are positioned on the substrate, the slight adhesive force of the solder paste being sufficient to keep them in place. The substrate is heated to the solder working temperature preferably by means of a controlled hot plate. The soldering process should be kept as short as possible: 10 to 15 seconds is sufficient to ensure good solder joints and evaporation of the binder fluid.
After soldering, the substrate must be cleaned of any remaining flux.

44-LEAD PLASTIC LEADED CHIP-CARRIER (PLCC); SOT-187A

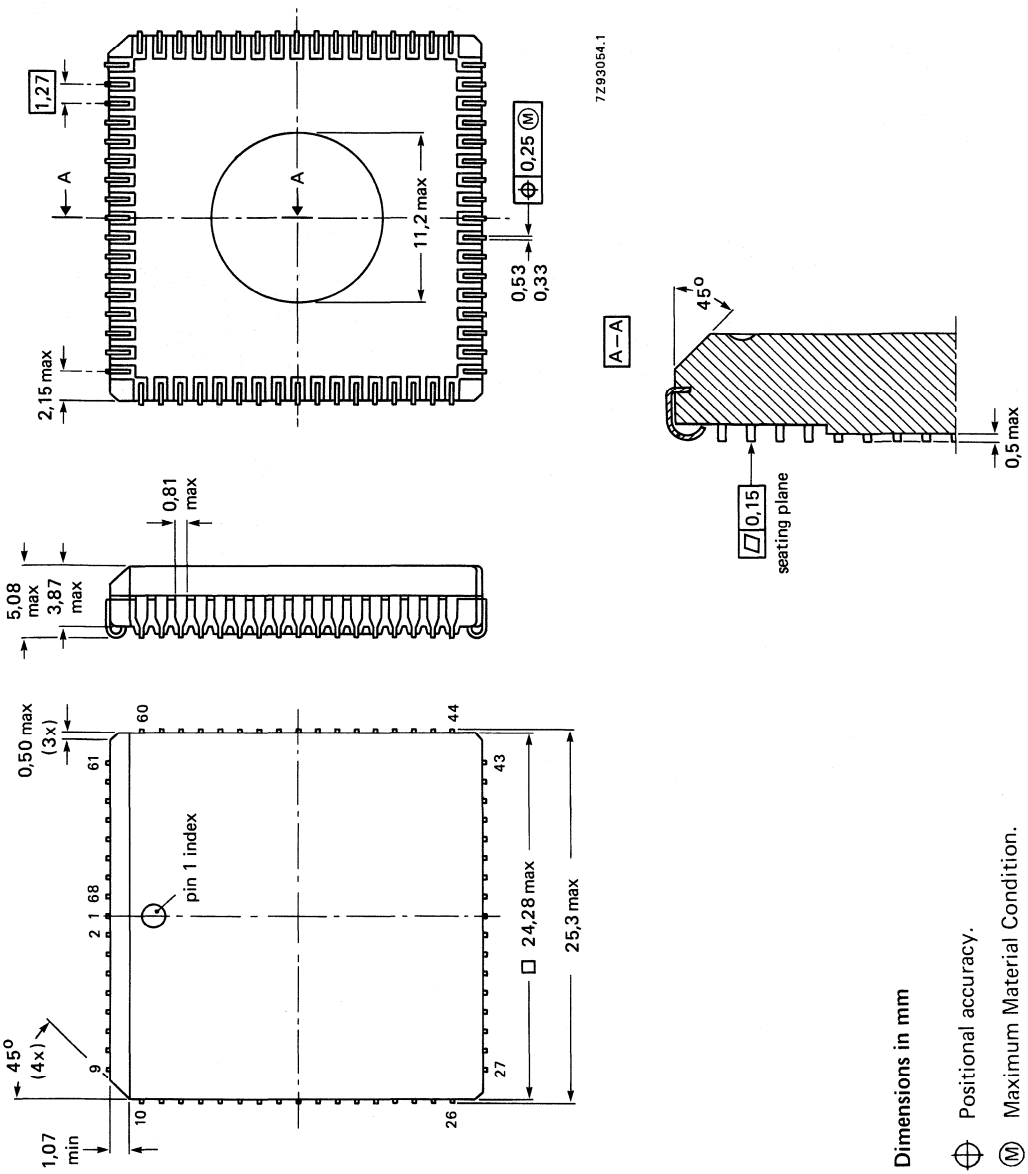


Dimensions in mm

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

68-LEAD PLASTIC LEADED-CHIP-CARRIER (PLCC); SOT-188A



Dimensions in mm

⊕ Positional accuracy.

Ⓜ Maximum Material Condition.

SOLDERING

1. By hand

Apply the soldering iron below the seating plane (or not more than 2 mm above it).
If its temperature is below 300 °C it must not be in contact for more than 10 seconds; if between 300 °C and 400 °C, for not more than 5 seconds.

2. By dip or wave

The maximum permissible temperature of the solder is 260 °C; this temperature must not be in contact with the joint for more than 5 seconds. The total contact time of successive solder waves must not exceed 5 seconds.
The device may be mounted up to the seating plane, but the temperature of the plastic body must not exceed the specified storage maximum. If the printed-circuit board has been pre-heated, forced cooling may be necessary immediately after soldering to keep the temperature within the permissible limit.

3. Repairing soldered joints

The same precautions and limits apply as in (1) above.

Electronic components and materials for professional, industrial and consumer uses from the world-wide Philips Group of Companies

Argentina: PHILIPS ARGENTINA S.A., Div. Elcoma, Vedia 3892, 1430 BUENOS AIRES, Tel. 541-7141/7242/7343/7444/7545.
Australia: PHILIPS INDUSTRIES HOLDINGS LTD., Elcoma Division, 11 Waltham Street, ARTARMON, N.S.W. 2064, Tel. (02)439 3322.
Austria: ÖSTERREICHISCHE PHILIPS BAUELEMENTE INDUSTRIE G.m.b.H., Triester Str. 64, A-1101 WIEN, Tel. 62911 11.
Belgium: N.V. PHILIPS & MBLE ASSOCIATED, 9 rue du Pavillon, B-1030 BRUXELLES, Tel. (02) 242 74 00.
Brazil: IBRAPE, Caixa Postal 7383, Av. Brigadeiro Faria Lima, 1735 SAO PAULO, SP, Tel. (011) 211-2600.
Canada: PHILIPS ELECTRONICS LTD., Elcoma Division, 601 Milner Ave., SCARBOROUGH, Ontario, M1B 1M8, Tel. 292-5161.
Chile: PHILIPS CHILENA S.A., Av. Santa Maria 0760, SANTIAGO, Tel. 39-4001.
Colombia: IND. PHILIPS DE COLOMBIA S.A., c/o IPRELENSO LTD., Cra. 21, No. 56-17, BOGOTA, D.E., Tel. 2497624.
Denmark: MINIWATT A/S, Strandlodsvej 2, P.O. Box 1919, DK 2300 COPENHAGEN S, Tel. (01) 54 11 33.
Finland: OY PHILIPS AB, Elcoma Division, Kaivokatu 8, SF-00100 HELSINKI 10, Tel. 17271.
France: R.T.C. LA RADIOTECHNIQUE-COMPELEC, 130 Avenue Ledru Rollin, F-75540 PARIS 11, Tel. 338 80-00.
Germany (Fed. Republic): VALVO, UB Bauelemente der Philips G.m.b.H., Valvo Haus, Burchardstrasse 19, D-2 HAMBURG 1, Tel. (040) 3296-0.
Greece: PHILIPS S.A. HELLENIQUE, Elcoma Division, 52, Av. Syngrou, ATHENS, Tel. 9215111.
Hong Kong: PHILIPS HONG KONG LTD., Elcoma Div., 15/F Philips Ind. Bldg., 24-28 Kung Yip St., KWAI CHUNG, Tel. (0)-245121.
India: PEICO ELECTRONICS & ELECTRICALS LTD., Elcoma Dept., Band Box Building, 254-D Dr. Annie Besant Rd., BOMBAY - 400025, Tel. 4220387/4220311.
Indonesia: P.T. PHILIPS-RALIN ELECTRONICS, Elcoma Div., Panim Bank Building, 2nd Fl., Jl. Jend. Sudirman, P.O. Box 223, JAKARTA, Tel. 716131.
Ireland: PHILIPS ELECTRICAL (IRELAND) LTD., Newstead, Clonskeagh, DUBLIN 14, Tel. 693355.
Italy: PHILIPS S.p.A., Sezione Elcoma, Piazza IV Novembre 3, I-20124 MILANO, Tel. 2-6752.1.
Japan: NIHON PHILIPS CORP., Shuwa Shinagawa Bldg., 26-33 Takanawa 3-chome, Minato-ku, TOKYO (108), Tel. 448-5611.
(IC Products) SIGNETICS JAPAN LTD., 8-7 Sanbancho Chiyoda-ku, TOKYO 102, Tel. (03) 230-1521.
Korea (Republic of): PHILIPS ELECTRONICS (KOREA) LTD., Elcoma Div., Philips House, 260-199 Itaewon-dong, Yongsan-ku, SEOUL, Tel. 794-4202.
Malaysia: PHILIPS MALAYSIA SDN. BERHAD, No. 4 Persiaran Barat, Petaling Jaya, P.O.B. 2163, KUALA LUMPUR, Selangor, Tel. 774411.
Mexico: ELECTRONICA, S.A de C.V., Carr. México-Toluca km. 62.5, TOLUCA, Edo. de México 50140, Tel. Toluca 91 (721) 613-00.
Netherlands: PHILIPS NEDERLAND, Marktgroep Elonco, Postbus 90050, 5600 PB EINDHOVEN, Tel. (040) 793333.
New Zealand: PHILIPS NEW ZEALAND LTD., Elcoma Division, 110 Mt. Eden Road, C.P.O. Box 1041, AUCKLAND, Tel. 605-914.
Norway: NORSK A/S PHILIPS, Electronica Dept., Sandstuveien 70, OSLO 6, Tel. 680200.
Peru: CADESA, Av. Alfonso Ugarte 1268, LIMA 5, Tel. 326070.
Philippines: PHILIPS INDUSTRIAL DEV. INC., 2246 Pasong Tamo, P.O. Box 911, Makati Comm. Centre, MAKATI-RIZAL 3116, Tel. 86-89-51 to 59.
Portugal: PHILIPS PORTUGUESA S.A.R.L., Av. Eng. Duarte Pacheco 6, 1009 LISBOA Codex, Tel. 683121.
Singapore: PHILIPS PROJECT DEV. (Singapore) PTE LTD., Elcoma Div., Lorong 1, Toa Payoh, SINGAPORE 1231, Tel. 3502000.
South Africa: EDAC (PTY.) LTD., 3rd Floor Rainer House, Upper Railway Rd. & Ove St., New Doornfontein, JOHANNESBURG 2001, Tel. 614-2362/9.
Spain: MINIWATT S.A., Balmes 22, BARCELONA 7, Tel. 301 63 12.
Sweden: PHILIPS KOMPONENTER A.B., Lidingövägen 50, S-11584 STOCKHOLM 27, Tel. 08/7821000.
Switzerland: PHILIPS A.G., Elcoma Dept., Allmendstrasse 140-142, CH-8027 ZÜRICH, Tel. 01-4882211.
Taiwan: PHILIPS TAIWAN LTD., 3rd Fl., San Min Building, 57-1, Chung Shan N. Rd, Section 2, P.O. Box 22978, TAIPEI, Tel. (02)-5631717.
Thailand: PHILIPS ELECTRICAL CO. OF THAILAND LTD., 283 Silom Road, P.O. Box 961, BANGKOK, Tel. 233-6330-9.
Turkey: TÜRK PHILIPS TICARET A.S., Elcoma Department, Inönü Cad. No. 78-80. ISTANBUL, Tel. 435910.
United Kingdom: MULLARD LTD., Mullard House, Torrington Place, LONDON WC1E 7HD, Tel. 01-5806633.
United States: (Active Devices & Materials) AMPEREX SALES CORP., Providence Pike, SLATERSVILLE, R.I. 02876, Tel. (401) 762-9000.
(Passive Devices) MEPCO/ELECTRA INC., Columbia Rd., MORRISTOWN, N.J. 07960, Tel. (201) 539-2000.
(Passive Devices & Electromechanical Devices) CENTRALAB INC., 5855 N. Glen Park Rd., MILWAUKEE, WI 53201, Tel. (414)228-7380.
(IC Products) SIGNETICS CORPORATION, 811 East Arques Avenue, SUNNYVALE, California 94086, Tel. (408) 739-7700.
Uruguay: LUZILECTRON S.A., Avda Uruguay 1287, P.O. Box 907, MONTEVIDEO, Tel. 914321.
Venezuela: IND. VENEZOLANAS PHILIPS S.A., c/o MAGNETICA S.A., Calle 6, Ed. Las Tres Jotas, App. Post. 78117, CARACAS, Tel. (02) 2393931

For all other countries apply to: Philips Electronic Components and Materials Division, International Business Relations, Building BAE, P.O. Box 218, 5600 MD EINDHOVEN, The Netherlands, Tel. +31 40 72 33 04, Telex 35000 phtcnl

A47

Printed in The Netherlands

9398 132 40011