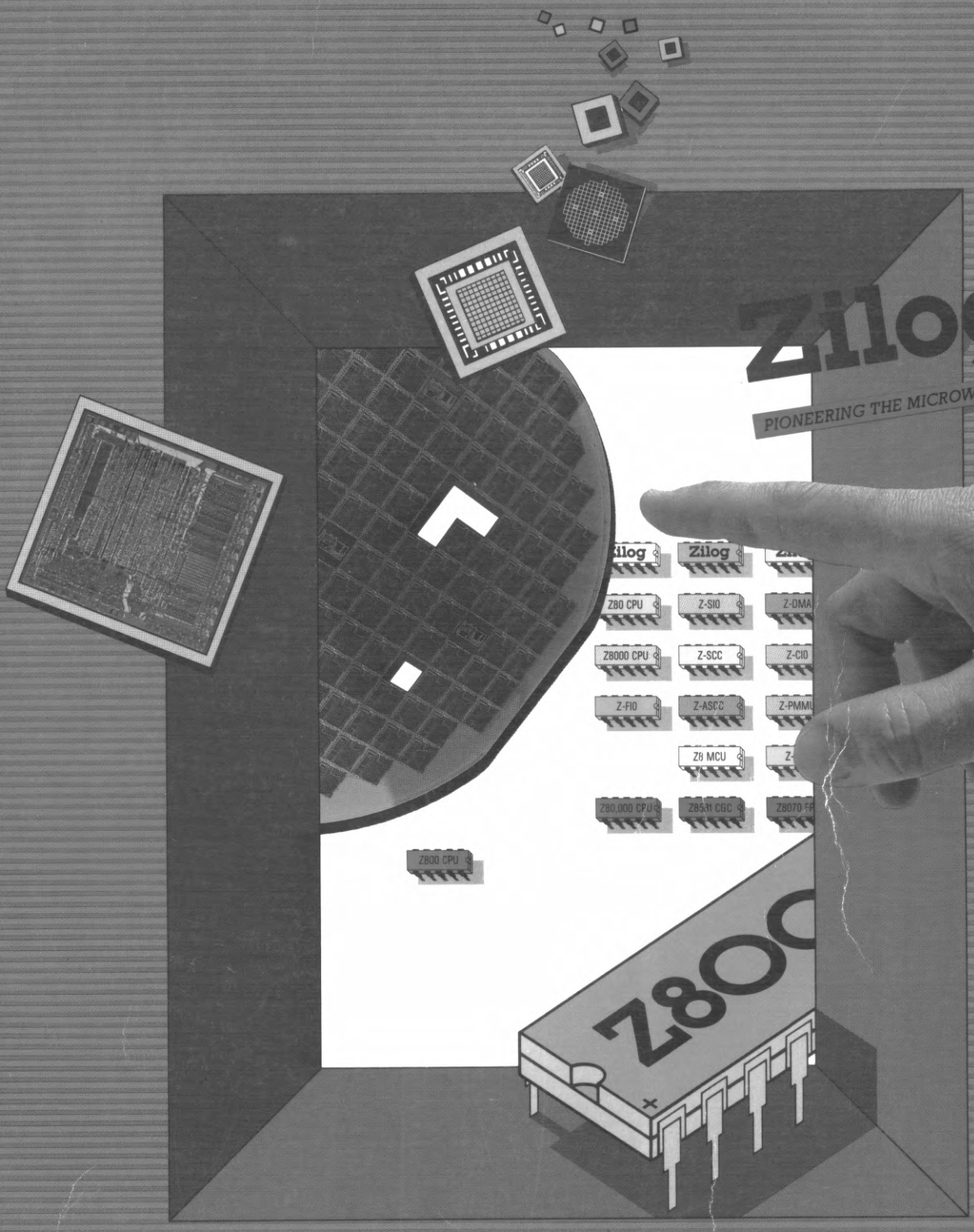


1983/84

COMPONENTS DATA BOOK

1983/84

Data Book

Zilog

Pioneering the Microworld

Microcomputers in Every Form

Zilog offers microprocessors in every form: from components and development systems to board-level products and complete general-purpose microcomputer systems. This edition of the **Zilog Data Book** describes Zilog components, development systems, and microcomputer boards. You'll also find a section on the in-depth training courses now offered for most Zilog products.

Zilog components, the basic building blocks for our other microcomputer products, include the 8-bit Z80® Microprocessor and its family of intelligent peripherals, the Z8™ Family of Single-Chip Microcomputers, and the 16-bit Z8000™ Microprocessor and its family of intelligent peripherals.

New products introduced in this data book are the Z800™ Family of Microprocessors, which continues the Z80® tradition, and the Z80,000™ Microprocessor, which establishes Zilog's supremacy in the 32-bit market. Also new is the Z8070 APU Arithmetic Processing Unit for the Z800, Z8000, and Z80,000 microprocessor families.

Zilog offers a wide variety of development environments, ranging from the inexpensive Z8 and Z8000 Development Modules to the more elaborate EMS 8000 Development System.

Also offered are Models 11, 21, and 31 of the System 8000 high-performance, time-sharing computer system. In addition, EMS 8000 and Z-SCAN 8000 both provide in-circuit emulation for the Z8000 Family of Microprocessors, and Z-SCAN 800 provides in-circuit emulation for the Z800 Family.

Table of Contents

Table of Contents (Continued)

Functional Index

Functional Index (Continued)

Part Number Index

Part Number Index (Continued)

*All 85XX components are compatible with processors other than Zilog's Z8001, Z8002, Z8003, and Z8004. For further information refer to the individual product specifications.

Z80

Family

Zilog

Pioneering the Microworld

Zilog Z80® Family

Sets the Industry Standard for 8 Bits

September 1983

Zilog remains an industry leader, thanks to continuing innovation in microcomputer concepts and integrated design as exemplified in the Z80 Family microcomputer products.

At Zilog, innovation means using proven, sophisticated mainframe and minicomputer concepts and translating them into the latest LSI technologies. Integration means more than designing an ever-greater number of functions onto a single chip. Zilog integrates technologies—LSI design enhanced by advances in computer-based system architecture and system design technologies.

Zilog offers microprocessor solutions to computing problems: from components and development systems to OEM board-level products and general-purpose microcomputer systems.

This guide to the Z80 Family of state-of-the-art microprocessors and intelligent peripheral controllers demonstrates Zilog's continued support for the Z80 microprocessor and the other members of the Z80 product family—a family first introduced in 1976 that continues to enjoy growing customer support while family chips are upgraded to newer and ever-higher standards.

The **Z8400 Z80 CPU Central Processing Unit** rapidly established itself as the most sophisticated, most powerful, and most versatile 8-bit microprocessor in the world. It offers many more features and functions than its competitor.

In addition to being source-code compatible with the 8080A microprocessor, the Z80 offers more instructions than the 8080A (158 vs. 78) and numerous other features that simplify hardware requirements and reduce programming effort while increasing throughput. The dual-register set of the Z80 CPU allows high-speed context switching and more efficient interrupt processing. Two index registers give additional memory-addressing flexibility and simplify the task of programming. Interfacing to dynamic memory is simplified by on-chip, programmable refresh logic. Block moves plus string- and bit-manipulation instructions reduce programming effort, program size, and execution time.

The new **Z80L Low-Power Family** widens the range of possible Z80 applications. Products in this family retain all the functions of the standard components while providing dramatic power savings and increased reliability. Available now in low-power versions are the Z80 CPU, Z80 CTC, Z80 PIO, and Z80 SIO.

The four traditional functions of a microcomputer system (parallel I/O, serial I/O, counting/timing, and direct memory access) are easily implemented by the Z80 CPU and the following well-proven family of Z80 peripheral devices: Z80 PIO, Z80 SIO, Z80 DART, Z80 CTC, and Z80 DMA.

The easily programmed, dual-channel **Z8420 Z80 PIO Parallel Input/Output Controller** offers two 8-bit I/O ports with individual handshake and pattern recognition logic. Both I/O ports operate in either a byte or a bit mode. In addition, this device can be programmed to generate interrupts for various status conditions.

All common data communications protocols, asynchronous as well as synchronous, are remarkably well handled by the **Z8440 Z80 SIO Serial Input/Output Controller**. This dual-channel receiver/transmitter device offers on-chip parity and CRC generation/checking. FIFO buffering and flag- and frame-detection generation logic are also offered.

If asynchronous-only applications are required, the cost-effective **Z8470 Z80 DART Dual Asynchronous Receiver/ Transmitter** can be used in place of the Z80 SIO. The Z80 DART offers all Z80 SIO asynchronous features in two channels.

Timing and event-counting functions are the forte of the **Z8430 Z80 CTC Counter/Timer Controller.** The CTC provides four counters, each with individually programmable prescalers. The CTC is a convenient source of programmable clock rates for the SIO.

With the **Z8410 Z80 DMA Direct Memory Access Controller,** data can be transferred directly between any two ports *(typically, I/O and memory). The* DMA transfers, searches, or search/transfers date in Byte-by-Byte, Burst, or Continuous modes. This device can achieve an impressive 2M bits per second data rate in the Search mode.

Z8400
Z80® CPU Central Processing Unit

Zilog

Product Specification

September 1983

Features

- The instruction set contains 158 instructions. The 78 instructions of the 8080A are included as a subset; 8080A software compatibility is maintained.
- Eight MHz, 6 MHz, 4 MHz and 2.5 MHz clocks for the Z80H, Z80B, Z80A, and Z80 CPU result in rapid instruction execution with consequent high data throughput.
- The extensive instruction set includes string, bit, byte, and word operations. Block searches and block transfers together with indexed and relative addressing result in the most powerful data handling capabilities in the microcomputer industry.
- The Z80 microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system may be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software through single-context switching, background-foreground programming, and single-level interrupt processing. In addition, two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high-speed interrupt processing: 8080 similar, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

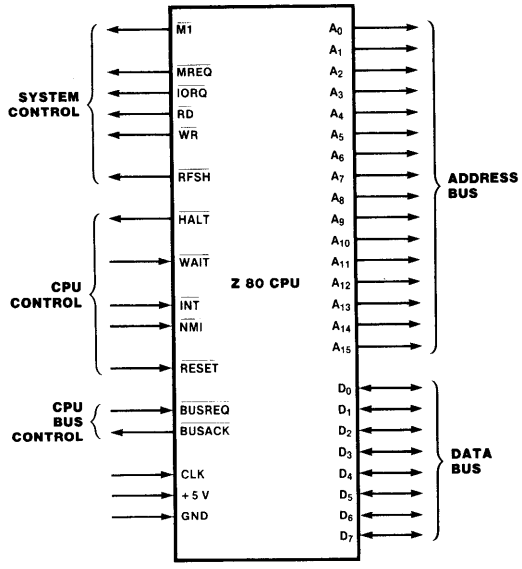


Figure 1. Pin Functions

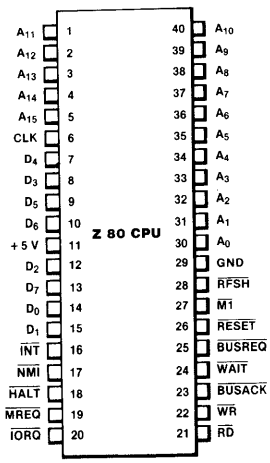


Figure 2. Pin Assignments

General Description

The Z80, Z80A, Z80B, and Z80H CPUs are third-generation single-chip microprocessors with exceptional computational power. They offer higher system throughput and more efficient memory utilization than comparable second-and third-generation microprocessors. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80 also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source. All output signals are fully decoded and timed to control standard memory or peripheral circuits, and it is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80 processors. Subsequent text provides more detail on the Z80 I/O controller family, registers, instruction set, interrupts and daisy chaining, and CPU timing.

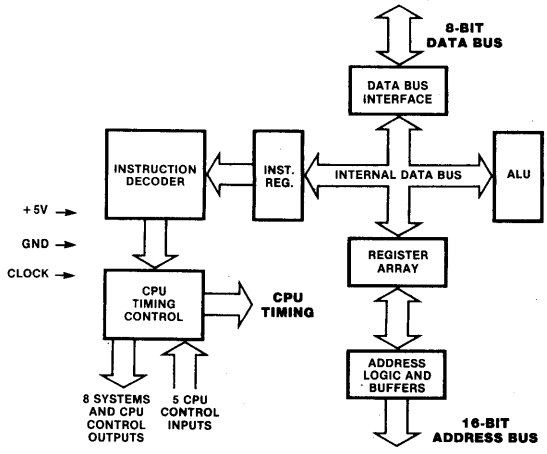


Figure 3. Z80 CPU Block Diagram

Z80 Microprocessor Family

The Zilog Z80 microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

Zilog has designed five components to provide extensive support for the Z80 microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Sync and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.

Z80 CPU Registers

Figure 4 shows three groups of registers within the Z80 CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

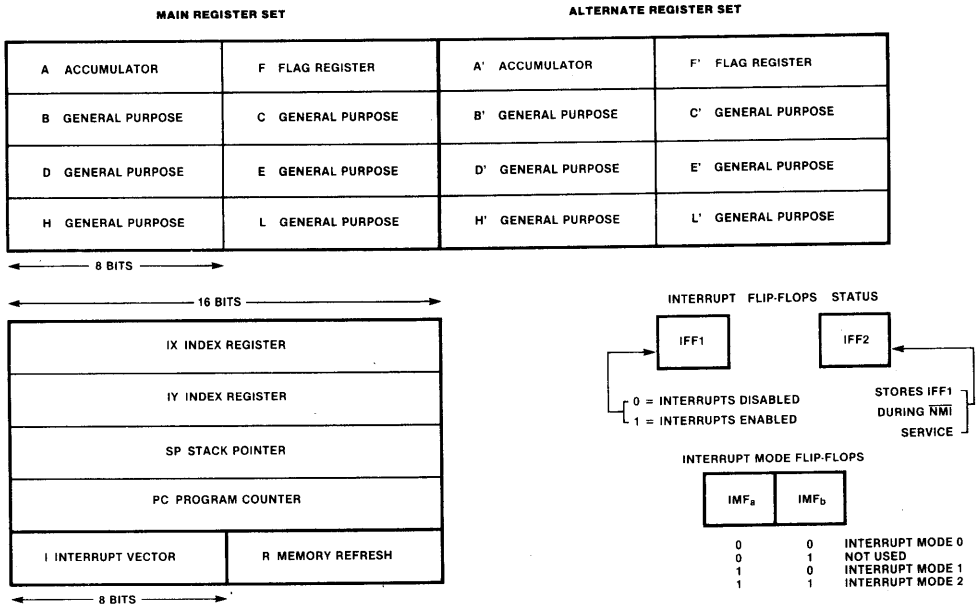


Figure 4. CPU Registers

Z80 CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Lower seven bits are automatically incremented and all eight are placed on the address bus during each instruction fetch cycle refresh time.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Holds address of the top of the stack. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80 CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{NMI}$ and $\overline{INT}$. The $\overline{NMI}$ is a non-maskable interrupt and has the highest priority. $\overline{INT}$ is a lower priority interrupt and it requires that interrupts be enabled in software in order to operate. $\overline{INT}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80 has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{INT}$, has three programmable response modes available. These are:

- Mode 0 — similar to the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt ($\overline{NMI}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted at all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routing.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80 response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request.

Mode 0 Interrupt Operation. This mode is similar to the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus. This is normally a Restart instruction, which will initiate a call to the selected one of eight restart locations in page zero of memory. Unlike the 8080, the Z80 CPU responds to the Call instruction with only one interrupt acknowledge cycle followed by two memory read cycles.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has a restart location of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80 microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit vector on the data bus during the interrupt acknowledge cycle. The CPU forms a pointer using this byte as the lower 8-bits and the contents of the I register as the upper 8-bits. This points to an entry in a table of addresses for interrupt service routines. The CPU then jumps to the routine at that address. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwired to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80 CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution	•	•	IFF_2 → Parity flag
LD A,R instruction execution	•	•	IFF_2 → Parity flag
Accept $\overline{NMI}$	0	IFF_1	IFF_1 → IFF_2 (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2	•	IFF_2 → IFF_1 at completion of an $\overline{NMI}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- ☐ 8-bit loads
- ☐ 16-bit loads
- ☐ Exchanges, block transfers, and searches
- ☐ 8-bit arithmetic and logic operations
- ☐ General-purpose arithmetic and CPU control
- ☐ 16-bit arithmetic operations
- ☐ Rotates and shifts
- ☐ Bit set, reset, and test operations
- ☐ Jumps
- ☐ Calls, returns, and restarts
- ☐ Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- ☐ Immediate
- ☐ Immediate extended
- ☐ Modified page zero
- ☐ Relative
- ☐ Extended
- ☐ Indexed
- ☐ Register
- ☐ Register indirect
- ☐ Implied
- ☐ Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	000 B 001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 101 ← d →	DD	3	5	19	011 E 100 H 101 L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	111 A
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10	
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19	
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19	
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13	
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9	
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9	
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9	
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9	

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.
IFF the content of the interrupt enable flip-flop, (IFF) is copied into the P/V flag.
For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables.

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD dd, nn	dd ← nn	•	•	X	X	•	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	X	•	•	•	11 011 101 DD 00 100 001 21 ← n → ← n →	4	4	14	
LD IY, nn	IY ← nn	•	•	X	X	•	•	•	11 111 101 FD 00 100 001 21 ← n → ← n →	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	X	X	•	•	•	00 101 010 2A ← n → ← n →	3	5	16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	•	•	X	X	•	•	•	11 101 101 ED 01 dd1 011 ← n → ← n →	4	6	20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	•	•	X	X	•	•	•	11 011 101 DD 00 101 010 2A ← n → ← n →	4	6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	X	X	•	•	•	11 111 101 FD 00 101 010 2A ← n → ← n →	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	X	X	•	•	•	00 100 010 22 ← n → ← n →	3	5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	X	X	•	•	•	11 101 101 ED 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	X	X	•	•	•	11 011 101 DD 00 100 010 22 ← n → ← n →	4	6	20	
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	X	X	•	•	•	11 111 101 FD 00 100 010 22 ← n → ← n →	4	6	20	
LD SP, HL	SP ← HL	•	•	X	X	•	•	•	11 111 001 F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	X	•	•	•	11 011 101 DD 11 111 001 F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	X	•	•	•	11 111 101 FD 11 111 001 F9	2	2	10	
PUSH qq	(SP − 2) ← qq_L (SP − 1) ← qq_H SP → SP − 2	•	•	X	X	•	•	•	11 qq0 101	1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP − 2) ← IX_L (SP − 1) ← IX_H SP → SP − 2	•	•	X	X	•	•	•	11 011 101 DD 11 100 101 E5	2	4	15	
PUSH IY	(SP − 2) ← IY_L (SP − 1) ← IY_H SP → SP − 2	•	•	X	X	•	•	•	11 111 101 FD 11 100 101 E5	2	4	15	
POP qq	qq_H ← (SP + 1) qq_L ← (SP) SP → SP + 2	•	•	X	X	•	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP → SP + 2	•	•	X	X	•	•	•	11 011 101 DD 11 100 001 E1	2	4	14	
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP → SP + 2	•	•	X	X	•	•	•	11 111 101 FD 11 100 001 E1	2	4	14	

NOTES: dd is any of the register pairs BC, DE, HL, SP.
qq is any of the register pairs AF, BC, DE, HL.
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively, e.g., BC_L = C, AF_H = A.

Exchange, Block Transfer, Block Search Groups

EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11 101 011 EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X	•	•	•	00 001 000 08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X	•	•	•	11 011 001 D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	•	•	X	•	X	•	•	•	11 100 011 E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 011 E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	•	•	X	0	X	①	0	•	11 101 101 ED 10 100 000 A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	①	0	•	11 101 101 ED 10 110 000 B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X 0 X	↕ ①	0	•	11 101 101 ED 10 101 000 A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X 0 X	0 ②	0	•	11 101 101 ED 10 111 000 B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 100 001 A1	2	4	16	
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 110 001 B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 101 001 A9	2	4	16	
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 111 001 B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.
② P/V flag is 0 at completion of instruction only.
③ Z flag is 1 if A = (HL), otherwise Z = 0.

8-Bit Arithmetic and Logical Group

ADD A, r	A ← A + r	↕	↕	X ↕ X	V	0	↕	10 [000] r	1	1	4	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X ↕ X	V	0	↕	11 [000] 110 ← n →	2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X ↕ X	V	0	↕	10 [000] 110	1	2	7	
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X ↕ X	V	0	↕	11 011 101 DD 10 [000] 110 ← d →	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X ↕ X	V	0	↕	11 111 101 FD 10 [000] 110 ← d →	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	X ↕ X	V	0	↕	[001]				s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above.
SUB s	A ← A − s	↕	↕	X ↕ X	V	1	↕	[010]				
SBC A, s	A ← A − s − CY	↕	↕	X ↕ X	V	1	↕	[011]				
AND s	A ← A ∧ s	↕	↕	X 1 X	P	0	0	[100]				
OR s	A ← A ∨ s	↕	↕	X 0 X	P	0	0	[110]				
XOR s	A ← A ⊕ s	↕	↕	X 0 X	P	0	0	[101]				
CP s	A − s	↕	↕	X ↕ X	V	1	↕	[111]				
INC r	r ← r + 1	↕	↕	X ↕ X	V	0	•	00 r [100]	1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X ↕ X	V	0	•	00 110 [100]	1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X ↕ X	V	0	•	11 011 101 DD 00 110 [100] ← d →	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X ↕ X	V	0	•	11 111 101 FD 00 110 [100] ← d →	3	6	23	
DEC m	m ← m − 1	↕	↕	X ↕ X	V	1	•	[101]				m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in opcode.

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands.	↕	↕	X	X	P	•	↕	00 100 111 27	1	1	4	Decimal adjust accumulator.
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111 2F	1	1	4	Complement accumulator (one's complement).
NEG	A – 0 – A	↕	↕	X	↕	X	V	1	↕	11 101 101 ED 01 000 100 44	2	2	8	Negate acc. (two's complement).
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111 3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111 37	1	1	4	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000 00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110 76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011 F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011 FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 110 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 ED 01 010 110 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 ED 01 011 110 5E	2	2	8	

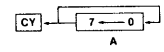
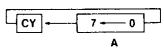
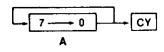
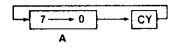
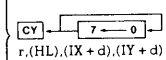
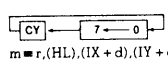
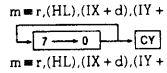
NOTES: IFF indicates the interrupt enable flip-flop.
CY indicates the carry flip-flop.
★ indicates interrupts are not sampled at the end of EI or DI.

16-Bit Arithmetic Group

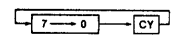
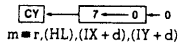
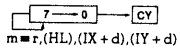
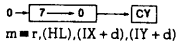
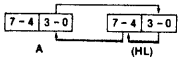
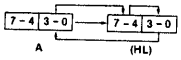
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001	1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 ED 01 ss1 010	2	4	15	
SBC HL, ss	HL ← HL – ss – CY	↕	↕	X	X	X	V	1	↕	11 101 101 ED 01 ss0 010	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 DD 01 pp1 001	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 FD 00 rr1 001	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011	1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 011 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 011 23	2	2	10	
DEC ss	ss ← ss – 1	•	•	X	•	X	•	•	•	00 ss1 011	1	1	6	
DEC IX	IX ← IX – 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 101 011 2B	2	2	10	
DEC IY	IY ← IY – 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 101 011 2B	2	2	10	

NOTES: ss is any of the register pairs BC, DE, HL, SP.
pp is any of the register pairs BC, DE, IX, SP.
rr is any of the register pairs BC, DE, IY, SP.

Rotate and Shift Group

RLCA		•	•	X	0	X	•	0	↕	00 000 111 07	1	1	4	Rotate left circular accumulator.
RLA		•	•	X	0	X	•	0	↕	00 010 111 17	1	1	4	Rotate left accumulator.
RRCA		•	•	X	0	X	•	0	↕	00 001 111 0F	1	1	4	Rotate right circular accumulator.
RRA		•	•	X	0	X	•	0	↕	00 011 111 1F	1	1	4	Rotate right accumulator.
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 r	2	2	8	Rotate left circular register r.
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 DD 11 001 011 CB – d – 00 000 110	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 FD 11 001 011 CB – d – 00 000 110	4	6	23	
RL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	010				Instruction format and states are as shown for RLC's. To form new opcode replace 000 or RLC's with shown code.
RRC m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	001				

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RR m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	011					
SLA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	100					
SRA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	101					
SRL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	111					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL). The content of the upper half of the accumulator is unaffected.
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

Bit Set, Reset and Test Group

BIT b, r	$Z \leftarrow r_b$	X	↕	X	1	X	X	0	•	11 001 011 CB 01 b r	2	2	8	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	X	↕	X	1	X	X	0	•	11 001 011 CB 01 b 110	2	3	12	
BIT b, (IX + d)$_b$	$Z \leftarrow \overline{(IX+d)}_b$	X	↕	X	1	X	X	0	•	11 011 101 DD 11 001 011 CB ← d → 01 b 110	4	5	20	
BIT b, (IY + d)$_b$	$Z \leftarrow \overline{(IY+d)}_b$	X	↕	X	1	X	X	0	•	11 111 101 FD 11 001 011 CB ← d → 01 b 110	4	5	20	b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 CB 11 b r	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 CB 11 b 110	2	4	15	
SET b, (IX + d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 DD 11 001 011 CB ← d → 11 b 110	4	6	23	
SET b, (IY + d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 FD 11 001 011 CB ← d → 11 b 110	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	10				To form new opcode replace 11 of SET b, s with 10. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) or location m.

Jump Group

JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 C3 ← n → ← n →	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →	3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 18 ← e-2 →	2	3	12	
JR C, e	If C = 0, continue If C = 1, PC ← PC + e	•	•	X	•	X	•	•	•	00 111 000 38 ← e-2 →	2 2	2 3	7 12	If condition not met. If condition is met.
JR NC, e	If C = 1, continue If C = 0, PC ← PC + e	•	•	X	•	X	•	•	•	00 110 000 30 ← e-2 →	2 2	2 3	7 12	If condition not met. If condition is met.
JP Z, e	If Z = 0 continue If Z = 1, PC ← PC + e	•	•	X	•	X	•	•	•	00 101 000 28 ← e-2 →	2 2	2 3	7 12	If condition not met. If condition is met.
JR NZ, e	If Z = 1, continue If Z = 0, PC ← PC + e	•	•	X	•	X	•	•	•	00 100 000 20 ← e-2 →	2 2	2 3	7 12	If condition not met. If condition is met.
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001 E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 DD 11 101 001 E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP (IY)	PC ← IY	•	•	X	X	•	•	•	11 111 101 FD 11 101 001 E9	2	2	8	
DJNZ, e	B ← B − 1 If B = 0, continue	•	•	X	X	•	•	•	00 010 000 10 ← e − 2 →	2	2	8	If B = 0.
	If B ≠ 0, PC ← PC + e									2	3	13	If B ≠ 0.

NOTES: e represents the extension in the relative addressing mode.
e is a signed two's complement number in the range < −126, 129 >.
e − 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

Call and Return Group

CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ PC ← nn	•	•	X	X	•	•	•	11 001 101 CD ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	X	•	•	•	11 cc 100 ← n → ← n →	3	3	10	If cc is false.
										3	5	17	If cc is true.
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$	•	•	X	X	•	•	•	11 001 001 C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	X	•	•	•	11 cc 000	1	1	5	If cc is false.
										1	3	11	If cc is true.
RETI	Return from interrupt	•	•	X	X	•	•	•	11 101 101 ED 01 001 101 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	X	•	•	•	11 101 101 ED 01 000 101 45	2	4	14	
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$	•	•	X	X	•	•	•	11 t 111	1	3	11	

cc	Condition
000	NZ non-zero
001	Z zero
010	NC non-carry
011	C carry
100	PO parity odd
101	PE parity even
110	P sign positive
111	M sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

NOTE: [1]RETN loads $IFF_2 \rightarrow IFF_1$

Input and Output Group

IN A, (n)	A ← (n)	•	•	X	X	•	•	•	11 011 011 DB ← n →	2	3	11	n to $A_0 \sim A_7$ Acc. to $A_8 \sim A_{15}$
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 ED 01 r 000	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL) ← (C) B ← B − 1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	X	11 101 101 ED 10 100 010 A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL) ← (C) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1 ②	X	X	X	X	1	X	11 101 101 ED 10 110 010 B2	2	5 (If B≠0)	21	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
											2	4 (If B=0)	16	
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X	↕ ①	X	X	X	X	1	X	11 101 101 ED 10 101 010 AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL) ← (C) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1 ②	X	X	X	X	1	X	11 101 101 ED 10 111 010 BA	2	5 (If B≠0)	21	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
											2	4 (If B=0)	16	
OUT (n), A	(n) ← A	•	•	X	X	•	•	•	11 010 011 D3 ← n →	2	3	11	n to $A_0 \sim A_7$ Acc. to $A_8 \sim A_{15}$
OUT (C), r	(C) ← r	•	•	X	X	•	•	•	11 101 101 ED 01 r 001	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C) ← (HL) B ← B − 1 HL ← HL + 1	X	↕ ①	X	X	X	X	1	X	11 101 101 ED 10 100 011 A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C) ← (HL) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1 ②	X	X	X	X	1	X	11 101 101 ED 10 110 011 B3	2	5 (If B≠0)	21	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
											2	4 (If B=0)	16	
OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1	X	↕ ①	X	X	X	X	1	X	11 101 101 ED 10 101 011 AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

NOTE: ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset.
② Z flag is set upon instruction completion only.

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1 ①	X	X	X	1	11 101 101 ED 10 111 011	2	5 (If B≠0) 4 (If B=0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

NOTE: ① Z flag is set upon instruction completion only.

Summary of Flag Operation

Instruction	D_7 S	Z	H	P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	V	0	↕	8-bit add or add with carry.
SUB s; SBC A, s; CP s; NEG	↕	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator.
AND s	↕	↕	1	P	0	0	Logical operations.
OR s, XOR s	↕	↕	0	P	0	0	
INC s	↕	↕	↕	V	0	•	8-bit increment.
DEC s	↕	↕	↕	V	1	•	8-bit decrement.
ADD DD, ss	•	•	X	•	0	↕	16-bit add.
ADC HL, ss	↕	↕	X	V	0	↕	16-bit add with carry.
SBC HL, ss	↕	↕	X	V	1	↕	16-bit subtract with carry.
RLA, RLCA, RRA; RRCA	•	•	0	•	0	↕	Rotate accumulator.
RL m; RLC m; RR m; RRC m; SLA m; SRA m; SRL m	↕	↕	0	P	0	↕	Rotate and shift locations.
RLD; RRD	↕	↕	0	P	0	•	Rotate digit left and right.
DAA	↕	↕	↕	P	•	↕	Decimal adjust accumulator.
CPL	•	•	1	•	1	•	Complement accumulator.
SCF	•	•	0	•	0	1	Set carry.
CCF	•	•	X	•	0	↕	Complement carry.
IN r (C)	↕	↕	0	P	0	•	Input register indirect.
INI, IND, OUTI; OUTD	X	↕	X	X	1	•	Block input and output. Z = 0 if B ≠ 0 otherwise Z = 0.
INIR; INDR; OTIR; OTDR	X	1	X	X	1	•	
LDI; LDD	X	X	0	↕	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LDIR; LDDR	X	X	0	0	0	•	
CPI; CPIR; CPD; CPDR	X	↕	X	↕	1	•	Block search instructions. Z = 1 if A = (HL), otherwise Z = 0. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LD A, I, LD A, R	↕	↕	0	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
BIT b, s	X	↕	1	X	0	•	The state of bit b of location s is copied into the Z flag.

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{\text{BUSACK}}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, $\overline{\text{RD}}$, and $\overline{\text{WR}}$ have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{\text{BUSREQ}}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{\text{NMI}}$ and is always recognized at the end of the current machine cycle. $\overline{\text{BUSREQ}}$ forces the CPU address bus, data bus, and control signals $\overline{\text{MREQ}}$, $\overline{\text{IORQ}}$, RD, and WR to go to a high-impedance state so that other devices can control these lines. $\overline{\text{BUSREQ}}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{\text{BUSREQ}}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{\text{HALT}}$. *Halt State* (output, active Low). $\overline{\text{HALT}}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{\text{INT}}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{\text{INT}}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{\text{IORQ}}$. *Input/Output Request* (output, active Low, 3-state). $\overline{\text{IORQ}}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{\text{IORQ}}$ is also generated concurrently with $\overline{\text{M1}}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{\text{M1}}$. *Machine Cycle One* (output, active Low). $\overline{\text{M1}}$, together with $\overline{\text{MREQ}}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. M1, together with IORQ, indicates an interrupt acknowledge cycle.

$\overline{\text{MREQ}}$. *Memory Request* (output, active Low, 3-state). $\overline{\text{MREQ}}$ indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{\text{NMI}}$. *Non-Maskable Interrupt* (input, negative edge-triggered). $\overline{\text{NMI}}$ has a higher priority than $\overline{\text{INT}}$. $\overline{\text{NMI}}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{\text{RD}}$. *Read* (output, active Low, 3-state). $\overline{\text{RD}}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{\text{RESET}}$. *Reset* (input, active Low). $\overline{\text{RESET}}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{\text{RESET}}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{\text{RFSH}}$. *Refresh* (output, active Low). $\overline{\text{RFSH}}$, together with $\overline{\text{MREQ}}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{\text{WAIT}}$. *Wait* (input, active Low). $\overline{\text{WAIT}}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{\text{WAIT}}$ periods can prevent the CPU from refreshing dynamic memory properly.

$\overline{\text{WR}}$. *Write* (output, active Low, 3-state). $\overline{\text{WR}}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

CPU Timing

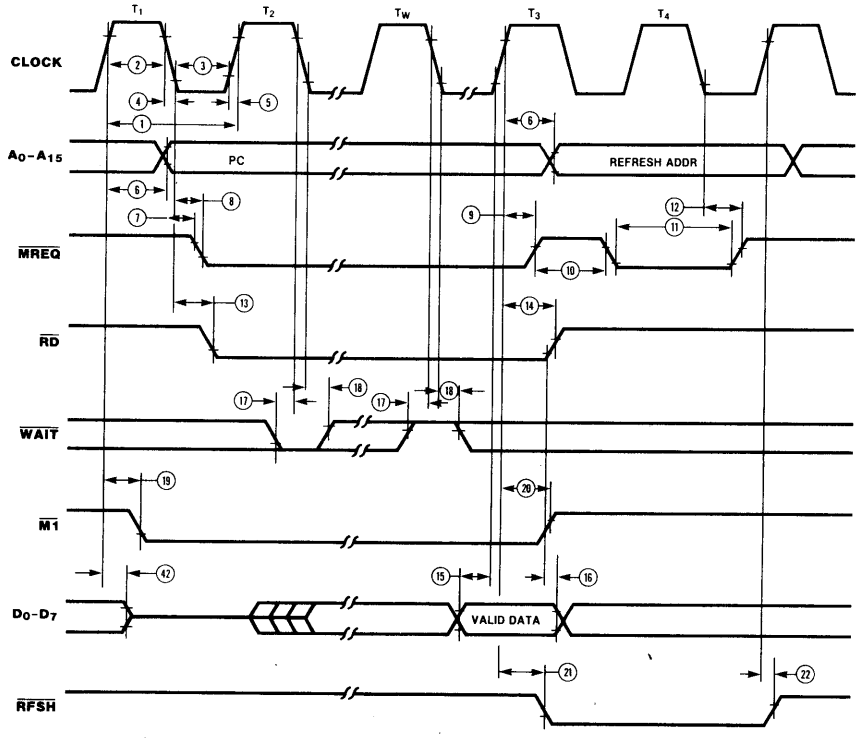
The Z80 CPU executes instructions by proceeding through a specific sequence of operations:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

The basic clock period is referred to as a T time or cycle, and three or more T cycles make up a machine cycle (M1, M2 or M3 for instance). Machine cycles can be extended either by the CPU automatically inserting one or more Wait states or by the insertion of one or more Wait states by the user.

Instruction Opcode Fetch. The CPU places the contents of the Program Counter (PC) on the address bus at the start of the cycle (Figure 5). Approximately one-half clock cycle later, $\overline{MREQ}$ goes active. When active, $\overline{RD}$ indicates that the memory data can be enabled onto the CPU data bus.

The CPU samples the $\overline{WAIT}$ input with the falling edge of clock state T_2. During clock states T_3 and T_4 of an $\overline{M1}$ cycle dynamic RAM refresh can occur while the CPU starts decoding and executing the instruction. When the Refresh Control signal becomes active, refreshing of dynamic memory can take place.



NOTE: T_W-Wait cycle added when necessary for slow ancilliary devices.

Figure 5. Instruction Opcode Fetch

CPU Timing (Continued)

Memory Read or Write Cycles. Figure 6 shows the timing of memory read or write cycles other than an opcode fetch ($\overline{M1}$) cycle. The $\overline{MREQ}$ and $\overline{RD}$ signals function exactly as in the fetch cycle. In a memory write cycle, $\overline{MREQ}$ also becomes active when the address bus is stable. The $\overline{WR}$ line is active when the data bus is stable, so that it can be used directly as an R/$\overline{W}$ pulse to most semiconductor memories.

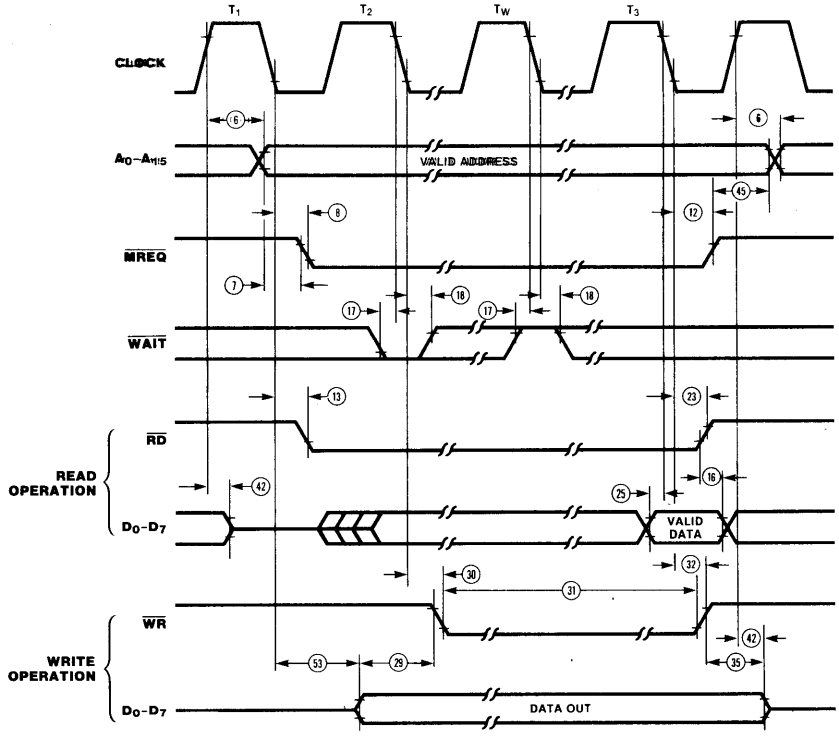
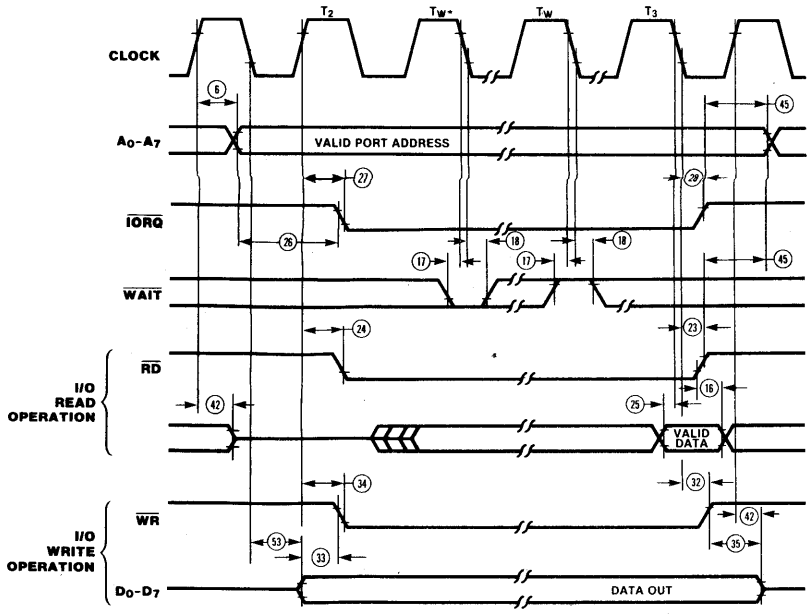


Figure 6. Memory Read or Write Cycles

CPU Timing (Continued)

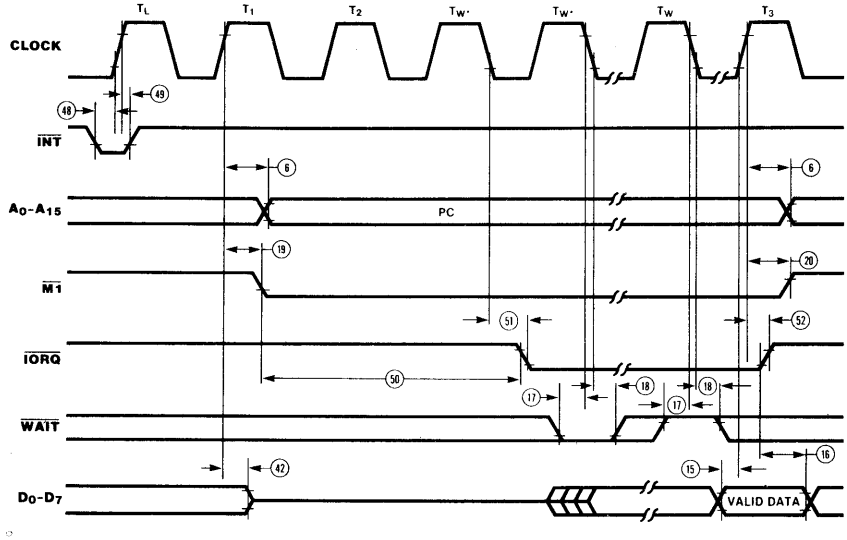
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_W). This extra Wait state allows sufficient time for an I/O port to decode the address from the port address lines.



NOTE: T_W* = One Wait cycle automatically inserted by CPU.

Figure 7. Input or Output Cycles

Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. The CPU automatically adds two Wait states to this cycle.



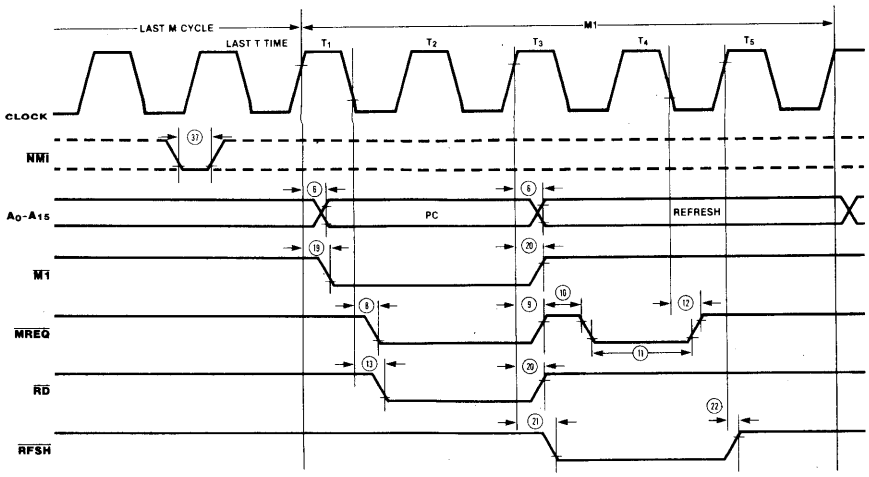
NOTE: 1) T_L = Last state of previous instruction.

2) Two Wait cycles automatically inserted by CPU(*).

Figure 8. Interrupt Request/Acknowledge Cycle

CPU Timing (Continued)

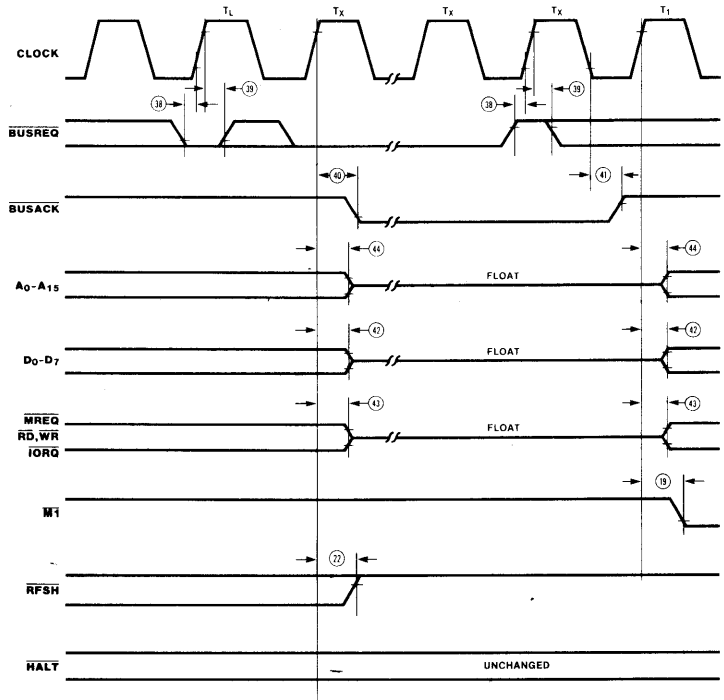
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal instruction fetch except that data put on the bus by the memory is ignored. The CPU instead executes a restart (RST) operation and jumps to the $\overline{NMI}$ service routine located at address 0066H (Figure 9).



*Although $\overline{NMI}$ is an asynchronous input, to guarantee its being recognized on the following machine cycle, $\overline{NMI}$'s falling edge must occur no later than the rising edge of the clock cycle preceding T_{LAST}.

Figure 9. Non-Maskable Interrupt Request Operation

Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to a high-impedance state with the rising edge of the next clock pulse. At that time, any external device can take control of these lines, usually to transfer data between memory and I/O devices.

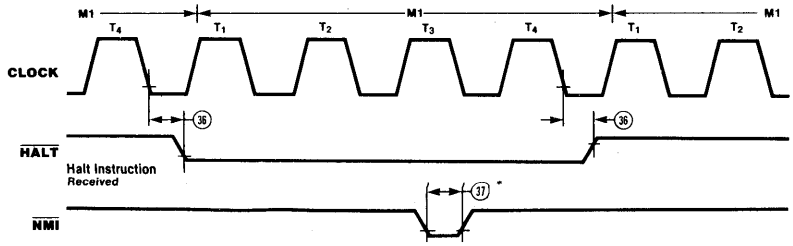


NOTE: T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Z-BUS Request/Acknowledge Cycle

CPU Timing (Continued)

Halt Acknowledge Cycle. When the CPU receives a Halt instruction, it executes NOP states until either an $\overline{INT}$ or $\overline{NMI}$ input is received. When in the Halt state, the $\overline{HALT}$ output is active and remains so until an interrupt is received (Figure 11).



NOTE: $\overline{INT}$ will also force a Halt exit.

*See note, Figure 9.

Figure 11. Halt Acknowledge Cycle

Reset Cycle. $\overline{RESET}$ must be active for at least three clock cycles for the CPU to properly accept it. As long as $\overline{RESET}$ remains active, the address and data buses float, and the control outputs are inactive. Once $\overline{RESET}$ goes inactive, three internal T cycles are consumed before the CPU resumes normal processing operation. $\overline{RESET}$ clears the PC register, so the first opcode fetch will be to location 0000 (Figure 12).

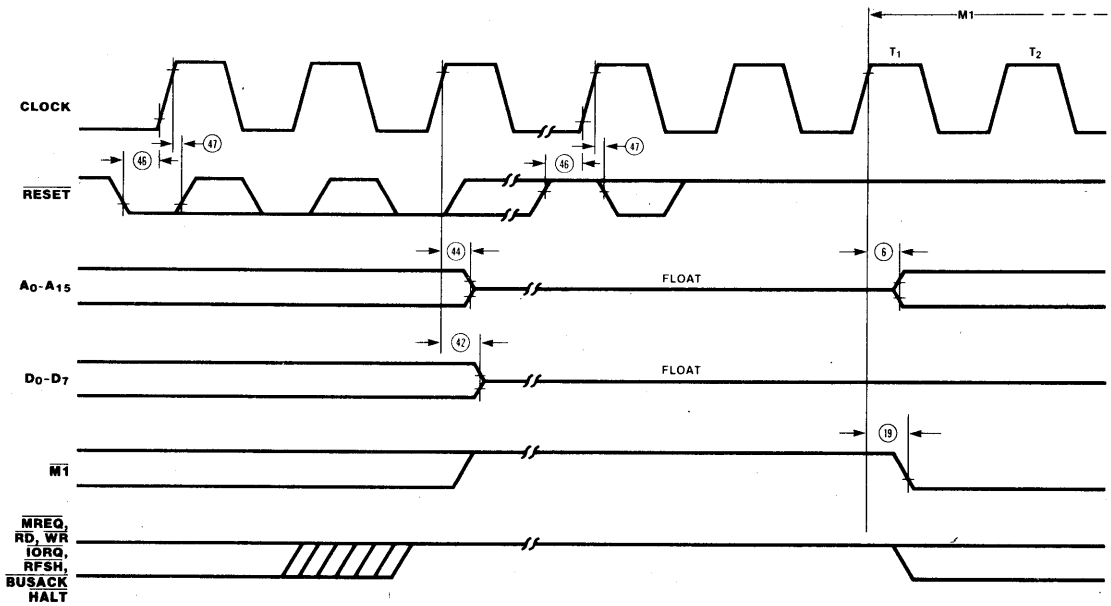


Figure 12. Reset Cycle

AC Characteristics

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU Min	Z80B CPU Max	Z80H CPU† Min	Z80H CPU† Max
1	TcC	Clock Cycle Time	400*		250*		165*		125*	
2	TwCh	Clock Pulse Width (High)	180*		110*		65*		55*	
3	TwCl	Clock Pulse Width (Low)	180	2000	110	2000	65	2000	55	2000
4	TfC	Clock Fall Time	—	30	—	30	—	20	—	10
5	TrC	Clock Rise Time	—	30	—	30	—	20	—	10
6	TdCr(A)	Clock ↑ to Address Valid Delay	—	145	—	110	—	90	—	80
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	125*	—	65*	—	35*	—	20*	—
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	100	—	85	—	70	—	60
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70	—	60
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	170*		110*		65*		45*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	360*	—	220*	—	135*	—	100*	—
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	100	—	85	—	70	—	60
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	130	—	95	—	80	—	70
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	100	—	85	—	70	—	60
15	TsD(Cr)	Data Setup Time to Clock ↑	50		35		30		30	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑	—	0	—	0	—	0	—	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	70	—	70	—	60	—	50	—
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	—	0	—	0	—	0	—	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	130	—	100	—	80	—	70
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay	—	130	—	100	—	80	—	70
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	180	—	130	—	110	—	95
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	150	—	120	—	100	—	85
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	110	—	85	—	70	—	60
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	100	—	85	—	70	—	60
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	60		50		40		30	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	320*	—	180*	—	110*	—	75*	—
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	90	—	75	—	65	—	55
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	110	—	85	—	70	—	60
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	190*	—	80*	—	25*	—	5*	—
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay	—	90	—	80	—	70	—	60
31	TwWR	$\overline{WR}$ Pulse Width	360*	—	220*	—	135*	—	100*	—
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	100	—	80	—	70	—	60
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	20*	—	-10*	—	-55*	—	55*	—
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	80	—	65	—	60	—	55
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	120*		60*		30*		15*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	300	—	300	—	260	—	225
37	TwNMI	$\overline{NMI}$ Pulse Width	80	—	80	—	70	—	60*	—
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	80	—	50	—	50	—	40	—

*For clock periods other than the minimums shown in the table, calculate parameters using the expressions in the table on the following page.

† Units in nanoseconds (ns). All timings are preliminary and subject to change.

AC Characteristics (Continued)

Number	Symbol	Parameter	Z80 CPU Min	Z80 CPU Max	Z80A CPU Min	Z80A CPU Max	Z80B CPU Min	Z80B CPU Max	Z80H CPU† Min	Z80H CPU† Max
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	—	0	—	0	—	0	—
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		120		100		90	—	80
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	—	110	—	100	—	90	—	80
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	90	—	90	—	80	—	70
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	110	—	80	—	70	—	60
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	110	—	90	—	80	—	70
45	TdCTr(A)	$\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑ to Address Hold Time	160*		80*		35*		20*	—
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	90	—	60	—	60	—	45	—
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	—	0	—	0	—	0	—	0
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	80	—	80	—	70	—	55	—
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	—	0	—	0	—	0	—	0
50	TdM1f(IORQf)	$\overline{M1}$ ↓ to $\overline{IORQ}$ ↓ Delay	920*		565*		365*		270*	—
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	110	—	85	—	70	—	60
52	TdCf(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	—	100	—	85	—	70	—	60
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	230	—	150	—	130	—	115

*For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC = TfC = 20 ns.
† Units in nanoseconds (ns). All timings are preliminary and subject to change.

Footnotes to AC Characteristics

Number	Symbol	Z80	Z80A	Z80B
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed
7	TdA(MREQf)	TwCh + TfC − 75	TwCh + TfC − 65	TwCh + TfC − 50
10	TwMREQh	TwCh + TfC − 30	TwCh + TfC − 20	TwCh + TfC − 20
11	TwMREQl	TcC − 40	TcC − 30	TcC − 30
26	TdA(IORQf)	TcC − 80	TcC − 70	TcC − 55
29	TdD(WRf)	TcC − 210	TcC − 170	TcC − 140
31	TwWR	TcC − 40	TcC − 30	TcC − 30
33	TdD(WRf)	TwCl + TrC − 180	TwCl + TrC − 140	TwCl + TrC − 140
35	TdWRr(D)	TwCl + TrC − 80	TwCl + TrC − 70	TwCl + TrC − 55
45	TdCTr(A)	TwCl + TrC − 40	TwCl + TrC − 50	TwCl + TrC − 50
50	TdM1f(IORQf)	2TcC + TwCh + TfC − 80	2TcC + TwCh + TfC − 65	2TcC + TwCh + TfC − 50

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
$V_{IHC} = V_{CC}$ -0.6 V
V_{ILC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ±0.5 V

Absolute Maximum Ratings

Storage Temperature . . . . . -65°C to +150°C
Temperature under Bias . . . . . Specified operating range
Voltages on all inputs and outputs with respect to ground . -0.3 V to +7 V
Power Dissipation . . . . . 1.5 W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

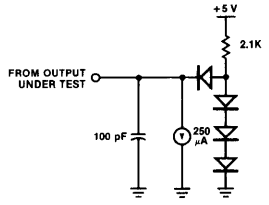
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = -40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = -55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a load capacitance of 100 pF. Add 10 ns delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus and 100 pF for address and control lines.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
I_{CC}	Power Supply Current				
	Z80		150[1]	mA	
	Z80A		200[2]	mA	
	Z80B		200	mA	
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float	-10	10[3]	µA	V_{OUT} = 0.4 to V_{CC}

1. For military grade parts, I_{CC} is 200 mA.
2. Typical rate for Z80A is 90 mA.
3. A_{15}-A_0, D_7-D_0, MREQ, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$.

Capacitance

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

T_A = 25°C, f = 1 MHz.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8400	CE	2.5 MHz	Z80 CPU (40-pin)	Z8400A	CMB	4.0 MHz	Z80A CPU (40-pin)
Z8400	CM	2.5 MHz	Same as above	Z8400A	CS	4.0 MHz	Same as above
Z8400	CMB	2.5 MHz	Same as above	Z8400A	DE	4.0 MHz	Same as above
Z8400	CS	2.5 MHz	Same as above	Z8400A	DS	4.0 MHz	Same as above
Z8400	DE	2.5 MHz	Same as above	Z8400A	PE	4.0 MHz	Same as above
Z8400	DS	2.5 MHz	Same as above	Z8400A	PS	4.0 MHz	Same as above
Z8400	PE	2.5 MHz	Same as above	Z8400B	CS	6.0 MHz	Z80B CPU (40-pin)
Z8400	PS	2.5 MHz	Same as above	Z8400B	DS	6.0 MHz	Same as above
Z8400A	CE	4.0 MHz	Z80A CPU (40-pin)	Z8400B	PS	6.0 MHz	Same as above
Z8400A	CM	4.0 MHz	Same as above				

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8410
Z80® DMA Direct Memory Access Controller

Zilog

Product Specification

September 1983

Features

- Transfers, searches and search/transfers in Byte-at-a-Time, Burst or Continuous modes. Cycle length and edge timing can be programmed to match the speed of any port.
- Dual port addresses (source and destination) generated for memory-to-I/O, memory-to-memory, or I/O-to-I/O operations. Addresses may be fixed or automatically incremented/decremented.
- Next-operation loading without disturbing current operations via buffered starting-address registers. An entire previous sequence can be repeated automatically.
- Extensive programmability of functions. CPU can read complete channel status.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic. Sophisticated, internally modifiable interrupt vectoring.
- Direct interfacing to system buses without external logic.

General Description

The Z-80 DMA (Direct Memory Access) is a powerful and versatile device for controlling and processing transfers of data. Its basic function of managing CPU-independent transfers between two ports is augmented by an array of features that optimize transfer speed and control with little or no external logic in systems using an 8- or 16-bit data bus and a 16-bit address bus.

Transfers can be done between any two ports (source and destination), including memory-to-I/O, memory-to-memory, and I/O-to-I/O. Dual port addresses are automatically generated for each transaction and may be either fixed or incrementing/decrementing. In addition, bit-maskable byte searches can be performed either concurrently with transfers or as an operation in itself.

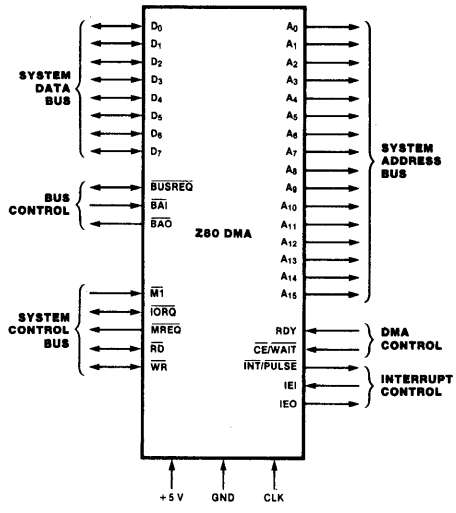


Figure 1. Pin Functions

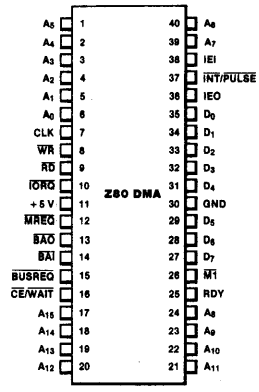


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 DMA contains direct interfacing to and independent control of system buses, as well as sophisticated bus and interrupt controls. Many programmable features, including variable cycle timing and auto-restart, minimize CPU software overhead. They are especially useful in adapting this special-purpose transfer processor to a broad variety of memory, I/O and CPU environments.

The Z-80 DMA is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 Family single-phase clock.

Functional Description

Classes of Operation. The Z-80 DMA has three basic classes of operation:

- Transfers of data between two ports (memory or I/O peripheral)
- Searches for a particular 8-bit maskable byte at a single port in memory or an I/O peripheral
- Combined transfers with simultaneous search between two ports

Figure 4 illustrates the basic functions served by these classes of operation.

During a transfer, the DMA assumes control of the system address and data buses. Data is read from one addressable port and written to the other addressable port, byte by byte. The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data may be written from one peripheral to another, from one area of main memory to another, or from a peripheral to main memory and vice versa.

During a search-only operation, data is read from the source port and compared byte by byte with a DMA-internal register containing a programmable match byte. This match byte may optionally be masked so that only certain bits within the match byte are compared. Search rates up to 1.25M bytes per second can be obtained with the 2.5 MHz Z-80 DMA or 2M bytes per second with the 4 MHz Z-80A DMA.

In combined searches and transfers, data is transferred between two ports while simultaneously searching for a bit-maskable byte match.

Data transfers or searches can be programmed to stop or interrupt under various conditions. In addition, CPU-readable status bits can be programmed to reflect the condition.

Modes of Operation. The Z-80 DMA can be programmed to operate in one of three transfer and/or search modes:

- *Byte-at-a-Time:* data operations are performed one byte at a time. Between each byte operation the system buses are released to the CPU. The buses are requested again for each succeeding byte operation.
- *Burst:* data operations continue until a port's Ready line to the DMA goes inactive. The DMA then stops and releases the system buses after completing its current byte operation.
- *Continuous:* data operations continue until the end of the programmed block of data is reached before the system buses are released. If a port's Ready line goes inactive before this occurs, the DMA simply pauses until the Ready line comes active again.

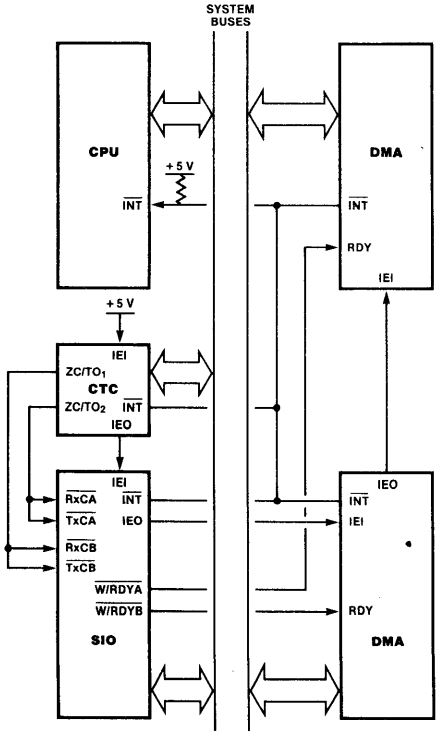
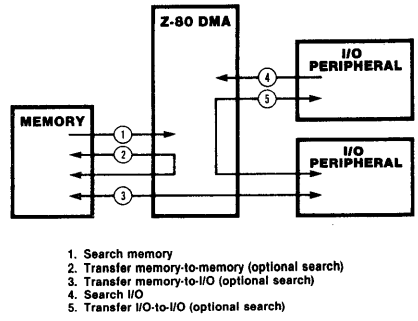


Figure 3. Typical Z-80 Environment



1. Search memory
2. Transfer memory-to-memory (optional search)
3. Transfer memory-to-I/O (optional search)
4. Search I/O
5. Transfer I/O-to-I/O (optional search)

Figure 4. Basic Functions of the Z-80 DMA

Functional Description (Continued)

In all modes, once a byte of data is read into the DMA, the operation on the byte will be completed in an orderly fashion, regardless of the state of other signals (including a port's Ready line).

Due to the DMA's high-speed buffered method of reading data, operations on one byte are not completed until the next byte is read in. This means that total transfer or search block lengths must be two or more bytes, and that block lengths programmed into the DMA must be one byte less than the desired block length (count is N-1 where N is the block length).

Commands and Status. The Z-80 DMA has several writable control registers and readable status registers available to the CPU. Control bytes can be written to the DMA whenever the DMA is not controlling the system buses, but the act of writing a control byte to the DMA disables the DMA until it is again enabled by a specific command. Status bytes can also be read at any such time, but writing the Read Status Byte command or the Initiate Read Sequence command disables the DMA.

Control bytes to the DMA include those which effect immediate command actions such as enable, disable, reset, load starting-address buffers, continue, clear counters, clear status bits and the like. In addition, many mode-setting control bytes can be written, including mode and class of operation, port configuration, starting addresses, block length, address counting rule, match and match-mask byte, interrupt conditions, interrupt vector, status-affects-vector condition, pulse counting, auto restart, Ready-line and Wait-line rules, and read mask.

Readable status registers include a general status byte reflecting Ready-line, end-of-block, byte-match and interrupt conditions, as well as 2-byte registers for the current byte count, Port A address and Port B address.

Variable Cycle. The Z-80 DMA has the unique feature of programmable operation-cycle length. This is valuable in tailoring the DMA to the particular requirements of other system components (fast or slow) and maximizes the data-transfer rate. It also eliminates external logic for signal conditioning.

There are two aspects to the variable cycle feature. First, the entire read and write cycles (periods) associated with the source and destination ports can be independently programmed as 2, 3 or 4 T-cycles long (more if Wait cycles are used), thereby increasing or decreasing the speed with which all DMA signals change (Figure 5).

Second, the four signals in each port specifically associated with transfers of data (I/O Request, Memory Request, Read, and Write) can each have its active trailing edge terminated one-half T-cycle early. This adds a further dimension of flexibility and speed, allowing such things as shorter-than-normal Read or Write signals that go inactive before data starts to change.

Address Generation. Two 16-bit addresses are generated by the Z-80 DMA for every transfer operation, one address for the source port and another for the destination port. Each address can be either variable or fixed. Variable addresses can increment or decrement from the programmed starting address. The fixed-address capability eliminates the need for separate enabling wires to I/O ports.

Port addresses are multiplexed onto the system address bus, depending on whether the DMA is reading the source port or writing to the destination port. Two readable address counters (2 bytes each) keep the current address of each port.

Auto Restart. The starting addresses of either port can be reloaded automatically at the end of a block. This option is selected by the Auto Restart control bit. The byte counter is cleared when the addresses are reloaded.

The Auto Restart feature relieves the CPU of software overhead for repetitive operations such as CRT refresh and many others. Moreover, when the CPU has access to the buses during byte-at-a-time or burst transfers, different starting addresses can be written into buffer registers during transfers, causing the Auto Restart to begin at a new location.

Interrupts. The Z-80 DMA can be programmed to interrupt the CPU on three conditions:

- Interrupt on Ready (before requesting bus)
- Interrupt on Match
- Interrupt on End of Block

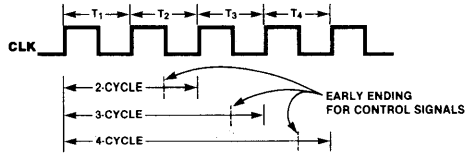


Figure 5. Variable Cycle Length

Functional Description (Continued)

Any of these interrupts cause an interrupt-pending status bit to be set, and each of them can optionally alter the DMA's interrupt vector. Due to the buffered constraint mentioned under "Modes of Operation," interrupts on Match at End of Block are caused by matches to the byte just prior to the last byte in the block.

The DMA shares the Z-80 Family's elaborate interrupt scheme, which provides fast interrupt service in real-time applications. In a Z-80 CPU environment, the DMA passes its internally modifiable 8-bit interrupt vector to the CPU, which adds an additional eight bits to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. In this process, CPU control is transferred directly to the interrupt routine, so that the next instruction executed after an interrupt acknowledge is the first instruction of the interrupt routine itself.

Pulse Generation. External devices can keep track of how many bytes have been transferred by using the DMA's pulse output, which provides a signal at 256-byte intervals. The interval sequence may be offset at the beginning by 1 to 255 bytes.

The Interrupt line outputs the pulse signal in a manner that prevents misinterpretation by the CPU as an interrupt request, since it only appears when the Bus Request and Bus Acknowledge lines are both active.

Pin Description

A_0-A_{15}. *System Address Bus* (output, 3-state). Addresses generated by the DMA are sent to both source and destination ports (main memory or I/O peripherals) on these lines.

$\overline{BAI}$. *Bus Acknowledge In* (input, active Low). Signals that the system buses have been released for DMA control. In multiple-DMA configurations, the $\overline{BAI}$ pin of the highest priority DMA is normally connected to the Bus Acknowledge pin of the CPU. Lower-priority DMAs have their $\overline{BAI}$ connected to the $\overline{BAO}$ of a higher-priority DMA.

$\overline{BAO}$. *Bus Acknowledge Out* (output, active Low). In a multiple-DMA configuration, this pin signals that no other higher-priority DMA has requested the system buses. $\overline{BAI}$ and $\overline{BAO}$ form a daisy chain for multiple-DMA priority resolution over bus control.

$\overline{BUSREQ}$. *Bus Request* (bidirectional, active Low, open drain). As an output, it sends requests for control of the system address bus, data bus and control bus to the CPU. As an input, when multiple DMAs are strung together in a priority daisy chain via $\overline{BAI}$ and $\overline{BAO}$, it senses when another DMA has requested the buses and causes this DMA to refrain from bus requesting until the other DMA is finished. Because it is a bidirectional pin, there cannot be any buffers between this DMA and any other DMA. It can, however, have a buffer between it and the CPU because it is unidirectional into the CPU. A pull-up resistor is connected to this pin.

$\overline{CE}$/$\overline{WAIT}$. *Chip Enable and Wait* (input, active Low). Normally this functions only as a $\overline{CE}$ line, but it can also be programmed to serve a $\overline{WAIT}$ function. As a $\overline{CE}$ line from the CPU, it becomes active when $\overline{WR}$ and $\overline{IORQ}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control or command bytes from the CPU to the DMA. As a $\overline{WAIT}$ line from memory or I/O devices, after the DMA has received a bus-request acknowledge from the CPU, it causes wait states to be inserted in the DMA's operation cycles thereby slowing the DMA to a speed that matches the memory or I/O device.

CLK. *System Clock* (input). Standard Z-80 single-phase clock at 2.5 MHz (Z-80 DMA) or 4.0 MHz (Z-80A DMA). For slower system clocks, a TTL gate with a pullup resistor may be adequate to meet the timing and voltage level specification. For higher-speed systems, use a clock driver with an active pullup to meet the V_{IH} specification and risetime requirements. In all cases there should be a resistive pullup to the power supply of 10K ohms (max) to ensure proper power when the DMA is reset.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). Commands from the CPU, DMA status, and data from memory or I/O peripherals are transferred on these lines.

IEI. *Interrupt Enable In* (input, active High). This is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this DMA. Thus, this signal blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

Pin Description (Continued)

$\overline{\text{INT}}$/PULSE. *Interrupt Request* (output, active Low, open drain). This requests a CPU interrupt. The CPU acknowledges the interrupt by pulling its $\overline{\text{IORQ}}$ output Low during an $\overline{\text{M1}}$ cycle. It is typically connected to the $\overline{\text{INT}}$ pin of the CPU with a pullup resistor and tied to all other $\overline{\text{INT}}$ pins in the system. This pin can also be used to generate periodic pulses to an external device. It can be used this way only when the DMA is bus master (i.e., the CPU's $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ lines are both Low and the CPU cannot see interrupts).

$\overline{\text{IORQ}}$. *Input/Output Request* (bidirectional, active Low, 3-state). As an input, this indicates that the lower half of the address bus holds a valid I/O port address for transfer of control or status bytes from or to the CPU, respectively; this DMA is the addressed port if its $\overline{\text{CE}}$ pin and its $\overline{\text{WR}}$ or $\overline{\text{RD}}$ pins are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates that the 8-bit or 16-bit address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both active simultaneously, an interrupt acknowledge is indicated.

$\overline{\text{M1}}$. *Machine Cycle One* (input, active Low). Indicates that the current CPU machine cycle is an instruction fetch. It is used by the DMA to decode the return-from-interrupt instruction (RETI) (ED-4D) sent by the CPU. During two-byte instruction fetches, $\overline{\text{M1}}$ is active as each opcode byte is fetched. An interrupt acknowledge is indicated when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active.

$\overline{\text{MREQ}}$. *Memory Request* (output, active Low, 3-state). This indicates that the address bus holds a valid address for a memory read or write operation. After the DMA has taken control of the system buses, it indicates a DMA transfer request from or to memory.

$\overline{\text{RD}}$. *Read* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to read status bytes from the DMA's read registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled read from a memory or I/O port address.

RDY. *Ready* (input, programmable active Low or High). This is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation. Depending on the mode of DMA operation (Byte, Burst or Continuous), the RDY line indirectly controls DMA activity by causing the $\overline{\text{BUSREQ}}$ line to go Low or High.

$\overline{\text{WR}}$. *Write* (bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to write control or command bytes to the DMA write registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled write to a memory or I/O port address.

Internal Structure

The internal structure of the Z-80 DMA includes driver and receiver circuitry for interfacing with an 8-bit system data bus, a 16-bit system address bus, and system control lines (Figure 6). In a Z-80 CPU environment, the DMA can be tied directly to the analogous pins on the CPU (Figure 7) with no additional buffering, except for the $\overline{\text{CE}}$/$\overline{\text{WAIT}}$ line.

The DMA's internal data bus interfaces with the system data bus and services all internal logic and registers. Addresses generated from this logic for Ports A and B (source and destination) of the DMA's single transfer channel are multiplexed onto the system address bus.

Specialized logic circuits in the DMA are dedicated to the various functions of external bus interfacing, internal bus control, byte matching, byte counting, periodic pulse generation, CPU interrupts, bus requests, and address generation. A set of twenty-one writable control registers and seven readable status registers provides the means by which the CPU governs and monitors the activities of these logic circuits. All registers are eight bits wide, with double-byte information stored in adjacent registers. The two address counters (two bytes each) for Ports A and B are buffered by the two starting addresses.

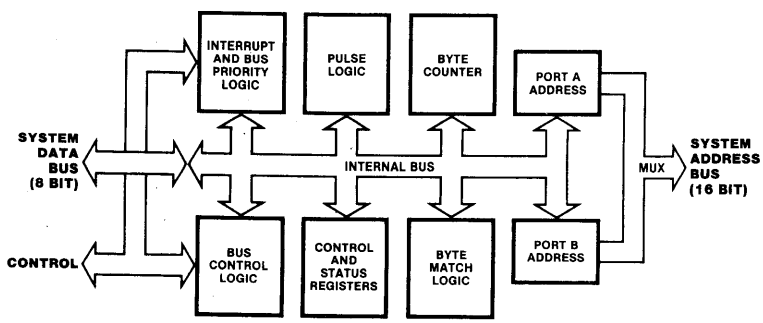


Figure 6. Block Diagram

Internal Structure (Continued)

The 21 writable control registers are organized into seven base-register groups, most of which have multiple registers. The base registers in each writable group contain both control/command bits and pointer bits that can be set to address other registers within the group. The seven readable status registers have no analogous second-level registers.

The registers are designated as follows, according to their base-register groups:

WR0-WR6 — Write Register groups 0 through 6 (7 base registers plus 14 associated registers)
RR0-RR6 — Read Registers 0 through 6

Writing to a register within a write-register group involves first writing to the base register, with the appropriate pointer bits set, then writing to one or more of the other registers within the group. All seven of the readable status registers are accessed sequentially according to a programmable mask contained in one of the writable registers. The section entitled "Programming" explains this in more detail.

A pipelining scheme is used for reading data in. The programmed block length is the number of bytes compared to the byte counter, which increments at the end of each cycle. In searches, data byte comparisons with the match byte are made during the read cycle of the next byte. Matches are, therefore, discovered only after the next byte is read in.

In multiple-DMA configurations, interrupt-request daisy chains are prioritized by the order in which their IEI and IEO lines are connected (Zilog Application Note 03-0041-01, *The Z-80 Family Program Interrupt Structure*). The system bus, however, may not be pre-empted. Any DMA that gains access to the system bus keeps the bus until it is finished.

Write Registers	
WR0	Base register byte Port A starting address (low byte) Port A starting address (high byte) Block length (low byte) Block length (high byte)
WR1	Base register byte Port A variable-timing byte
WR2	Base register byte Port B variable-timing byte
WR3	Base register byte Mask byte Match byte
WR4	Base register byte Port B starting address (low byte) Port B starting address (high byte) Interrupt control byte Pulse control byte Interrupt vector
WR5	Base register byte
WR6	Base register byte Read mask

Read Registers	
RR0	Status byte
RR1	Byte counter (low byte)
RR2	Byte counter (high byte)
RR3	Port A address counter (low byte)
RR4	Port A address counter (high byte)
RR5	Port B address counter (low byte)
RR6	Port B address counter (high byte)

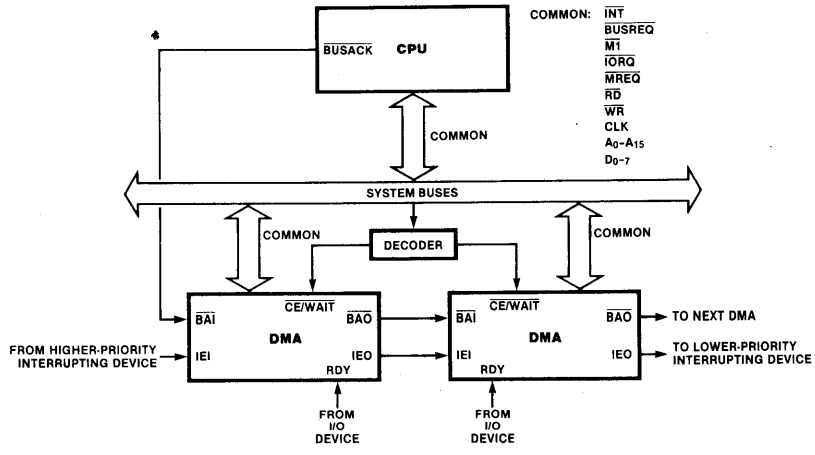


Figure 7. Multiple-DMA Interconnection to the Z-80 CPU

Programming The Z-80 DMA has two programmable fundamental states: (1) an enabled state, in which it can gain control of the system buses and direct the transfer of data between ports, and (2) a disabled state, in which it can initiate neither bus requests nor data transfers. When the DMA is powered up or reset by any means, it is automatically placed into the disabled state. Program commands can be written to it by the CPU in either state, but this automatically puts the DMA in the disabled state, which is maintained until an enable command is issued by the CPU. The CPU must program the DMA in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using an Output instruction (such as OTIR for the Z-80 CPU).

Writing. Control or command bytes are written into one or more of the Write Register groups (WR0-WR6) by first writing to the base register byte in that group. All groups have base registers and most groups have additional associated registers. The associated registers in a group are sequentially accessed by first writing a byte to the base register containing register-group identification and pointer bits (1's) to one or more of that base register's associated registers.

This is illustrated in Figure 8b. In this figure, the sequence in which associated registers within a group can be written to is shown by the vertical position of the associated registers. For example, if a byte written to the DMA contains the bits that identify WR0 (bits D0, D1 and D7), and also contains 1's in the bit positions that point to the associated "Port A Starting Address (low byte)" and "Port A Starting Address (high byte)," then the next two bytes written to the DMA will be stored in these two registers, in that order.

Reading. The Read Registers (RR0-RR6) are read by the CPU by addressing the DMA as an I/O port using an Input instruction (such as INIR for the Z-80 CPU). The readable bytes contain DMA status, byte-counter values, and port addresses since the last DMA reset. The registers are always read in a fixed sequence beginning with RR0 and ending with RR6. However, the register read in this sequence is determined by programming the Read Mask in WR6. The sequence of reading is initialized by writing an Initiate Read Sequence or Set Read Status command to WR6. After a Reset DMA, the sequence must be initialized with the Initiate Read Sequence command or a Read Status command. The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or Read Status command.

Fixed-Address Programming. A special circumstance arises when programming a destination port to have a fixed address. The load command in WR6 only loads a fixed address to a port selected as the source, not to a port selected as the destination. Therefore, a fixed destination address must be loaded by temporarily declaring it a fixed-source address and subsequently declaring the true source as such, thereby implicitly making the other a destination.

The following example illustrates the steps in this procedure, assuming that transfers are to occur from a variable-address source (Port A) to a fixed-address destination (Port B):

1. Temporarily declare Port B as source in WR0.
2. Load Port B address in WR6.
3. Declare Port A as source in WR0.
4. Load Port A address in WR6.
5. Enable DMA in WR6.

Figure 9 illustrates a program to transfer data from memory (Port A) to a peripheral device (Port B). In this example, the Port A memory starting address is 1050_H and the Port B peripheral fixed address is 05_H. Note that the data flow is 1001_H bytes—one more than specified by the block length. The table of DMA commands may be stored in consecutive memory locations and transferred to the DMA with an output instruction such as the Z-80 CPU's OTIR instruction.

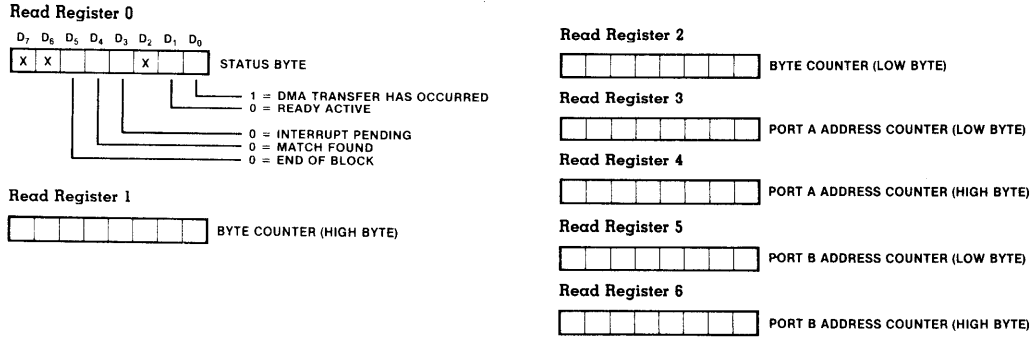


Figure 8a. Read Registers

Programming
(Continued)

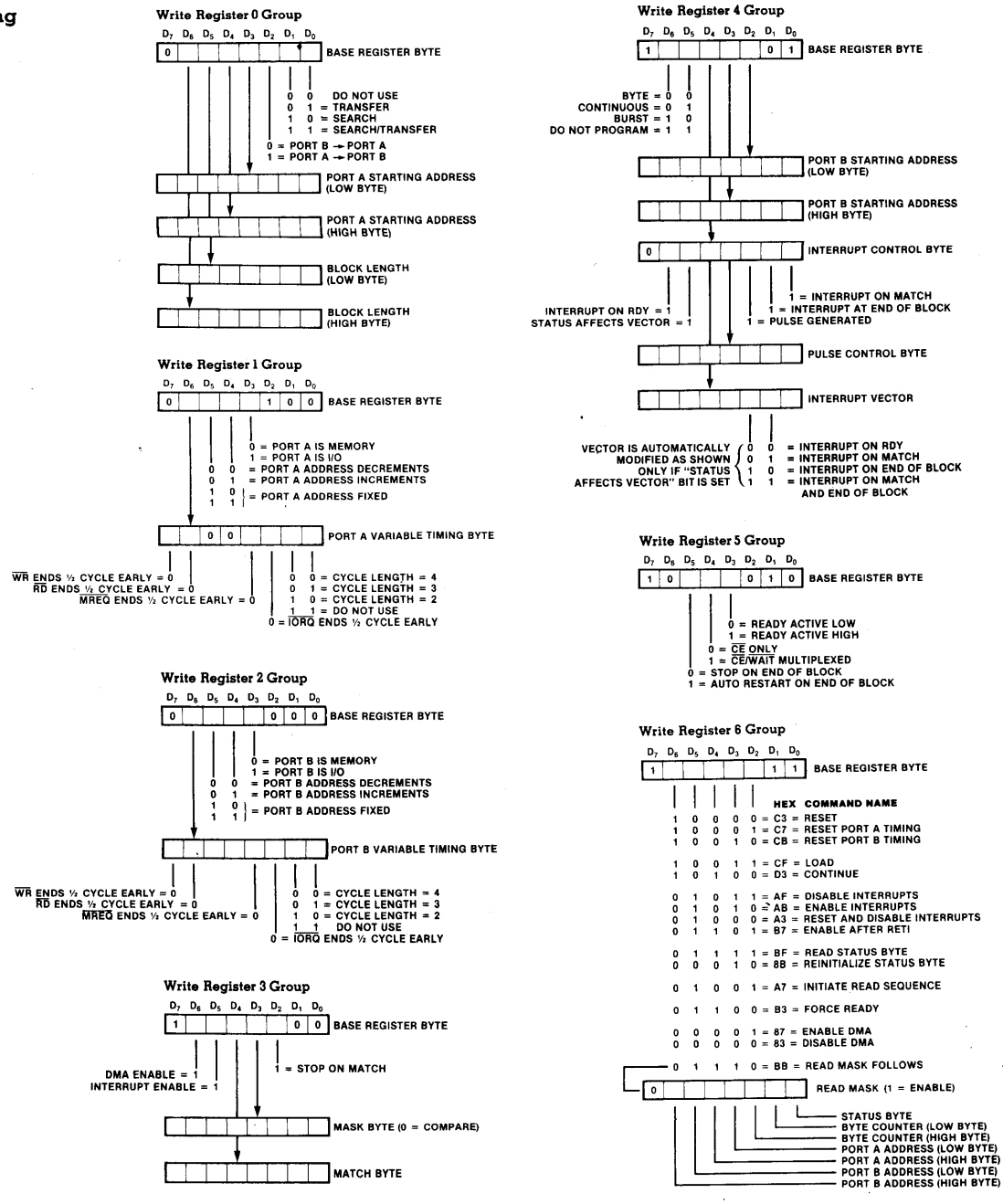


Figure 8b. Write Registers

Comments	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HEX
WR0 sets DMA to receive block length, Port A starting address and temporarily sets Port B as source.	0	1 Block Length Upper Follows	1 Block Length Lower Follows	1 Port A Upper Address Follows	1 Port A Lower Address Follows	0 B → A Temporary for Loading B Address*	0	1 Transfer, No Search	79
Port A address (lower)	0	1	0	1	0	0	0	0	50
Port A address (upper)	0	0	0	1	0	0	0	0	10
Block length (lower)	0	0	0	0	0	0	0	0	00
Block length (upper)	0	0	0	1	0	0	0	0	10
WR1 defines Port A as memory with fixed incrementing address.	0	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory	1	0	0	14
WR2 defines Port B as peripheral with fixed address.	0	0 No Timing Follows	1 Fixed Address	0	1 Port is I/O	0	1	0	28
WR4 sets mode to Burst, sets DMA to expect Port B address.	1	1 Burst Mode	0	0 No Interrupt Control Byte Follows	0 No Upper Address	1 Port B Lower Address Follows	0	1	C5
Port B address (lower)	0	0	0	0	0	1	0	1	05
WR5 sets Ready active High.	1	0	0 No Auto Restart	0 No Wait States	1 RDY Active High	0	1	0	8A
WR6 loads Port B address and resets block counter.*	1	1	0	0	1	1	1	1	CF
WR0 sets Port A as source.*	0	0	0 No Address or Block Length Bytes	0	0	1 A → B	0 Transfer, No Search	1	05
WR6 loads Port A address and resets block counter.	1	1	0	0	1	1	1	1	CF
WR6 enables DMA to start operation.	1	0	0	0	0	1	1	1	87

NOTE: The actual number of bytes transferred is one more than specified by the block length.
*These entries are necessary only in the case of a fixed destination address.

Figure 9. Sample DMA Program

Inactive State Timing (DMA as CPU Peripheral)

In its disabled or inactive state, the DMA is addressed by the CPU as an I/O peripheral for write and read (control and status) operations. Write timing is illustrated in Figure 10.

Reading of the DMA's status byte, byte counter or port address counters is illustrated in Figure 11. These operations require less than three T-cycles. The $\overline{CE}$, $\overline{IORQ}$ and $\overline{RD}$ lines are made active over two rising edges of CLK, and data appears on the bus approximately one T-cycle after they become active.

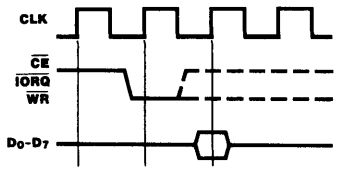


Figure 10. CPU-to-DMA Write Cycle

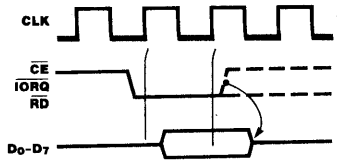


Figure 11. CPU-to-DMA Read Cycle

Active State Timing (DMA as Bus Controller)

Default Read and Write Cycles. By default, and after reset, the DMA's timing of read and write operations is exactly the same as the Z-80 CPU's timing of read and write cycles for memory and I/O peripherals, with one exception: during a read cycle, data is latched on the falling edge of T_3 and held on the data bus across the boundary between read and write cycles, through the end of the following write cycle.

Figure 12 illustrates the timing for memory-to-I/O port transfers and Figure 13 illustrates I/O-to-memory transfers. Memory-to-memory and I/O-to-I/O transfer timings are simply permutations of these diagrams.

The default timing uses three T-cycles for memory transactions and four T-cycles for I/O transactions, which include one automatically inserted wait cycle between T_2 and T_3. If the $\overline{CE/WAIT}$ line is programmed to act as a $\overline{WAIT}$ line during the DMA's active state, it is sampled on the falling edge of T_2 for memory transactions and the falling edge of T_W for I/O transactions. If $\overline{CE/WAIT}$ is Low during this time another T-cycle is added, during which the $\overline{CE/WAIT}$ line will again be sampled. The duration of transactions can thus be indefinitely extended.

Variable Cycle and Edge Timing. The Z-80 DMA's default operation-cycle length for the source (read) port and destination (write) port can be independently programmed. This variable-cycle feature allows read or write cycles consisting of two, three or four T-cycles (more if Wait cycles are inserted), thereby increasing or decreasing the speed of all signals generated by the DMA. In addition,

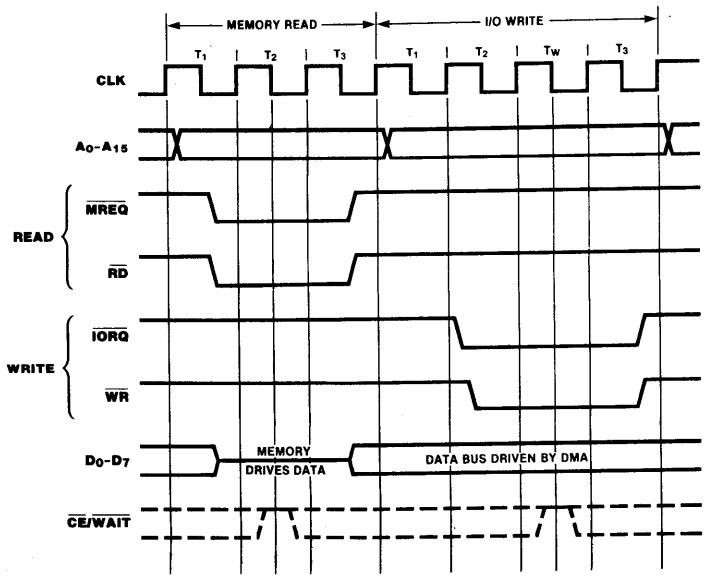


Figure 12. Memory-to-I/O Transfer

Active State Timing (DMA as Bus Controller) (Continued)

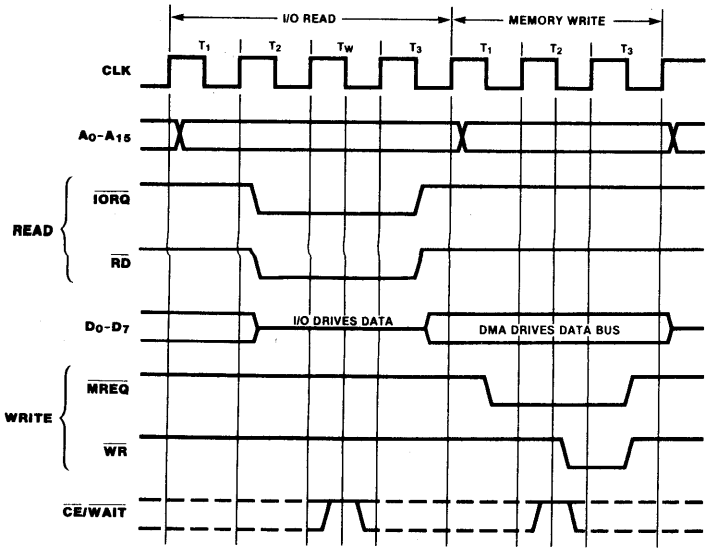


Figure 13. I/O-to-Memory Transfer

the trailing edges of the $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$ and $\overline{WR}$ signals can be independently terminated one-half cycle early. Figure 14 illustrates this.

In the variable-cycle mode, unlike default timing, $\overline{IORQ}$ comes active one-half cycle before $\overline{MREQ}$, $\overline{RD}$ and $\overline{WR}$. $\overline{CE}/\overline{WAIT}$ can be used to extend only the 3 or 4 T-cycle variable memory cycles and only the 4-cycle variable I/O cycle. The $\overline{CE}/\overline{WAIT}$ line is sampled at the falling edge of T_2 for 3- or 4-cycle memory cycles, and at the falling edge of T_3 for 4-cycle I/O cycles.

During transfers, data is latched on the clock edge causing the rising edge of $\overline{RD}$ and held through the end of the write cycle.

Bus Requests. Figure 15 illustrates the bus request and acceptance timing. The RDY line, which may be programmed active High or Low, is sampled on every rising edge of CLK. If it is found to be active, and if the bus is not in use by any other device, the following rising edge of CLK drives $\overline{BUSREQ}$ low. After receiving $\overline{BUSREQ}$ the CPU acknowledges on the $\overline{BAI}$ input either directly or through a multiple-DMA daisy chain. When a Low is detected on $\overline{BAI}$ for two consecutive rising edges of CLK, the DMA will begin transferring data on the next rising edge of CLK.

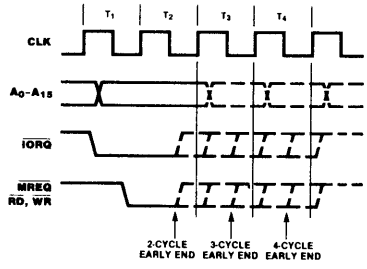


Figure 14. Variable-Cycle and Edge Timing

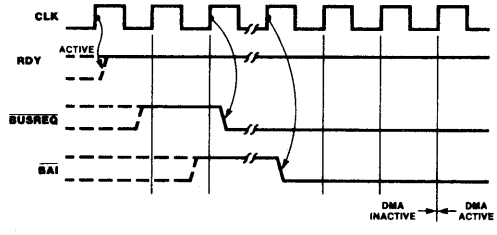


Figure 15. Bus Request and Acceptance

Active State Timing (DMA as Bus Controller) (Continued)

Bus Release Byte-at-a-Time. In Byte-at-a-Time mode, $\overline{BUSREQ}$ is brought High on the rising edge of CLK prior to the end of each read cycle (search-only) or write cycle (transfer and transfer/search) as illustrated in Figure 16. This is done regardless of the state of RDY. There is no possibility of confusion when a Z-80 CPU is used since the CPU cannot begin an operation until the following T-cycle. Most other CPUs are not bothered by this either, although note should be taken of it. The next bus request for the next byte will come after both $\overline{BUSREQ}$ and $\overline{BAI}$ have returned High.

Bus Release at End of Block. In Burst and Continuous modes, an end of block causes $\overline{BUSREQ}$ to go High usually on the same rising edge of CLK in which the DMA completes the transfer of the data block (Figure 17). The last byte in the block is transferred even if RDY goes inactive before completion of the last byte transfer.

Bus Release on Not Ready. In Burst mode, when RDY goes inactive it causes $\overline{BUSREQ}$ to go High on the next rising edge of CLK after the completion of its current byte operation (Figure 18). The action on $\overline{BUSREQ}$ is thus somewhat delayed from action on the RDY line. The DMA always completes its current byte operation in an orderly fashion before releasing the bus.

By contrast, $\overline{BUSREQ}$ is not released in Continuous mode when RDY goes inactive. Instead, the DMA idles after completing the current byte operation, awaiting an active RDY again.

Bus Release on Match. If the DMA is programmed to stop on match in Burst or Continuous modes, a match causes $\overline{BUSREQ}$ to go inactive on the next DMA operation, i.e., at the end of the next read in a search or at the end of the following write in a transfer (Figure 19). Due to the pipelining scheme, matches are determined while the next DMA read or write is being performed.

The RDY line can go inactive after the matching operation begins without affecting this bus-release timing.

Interrupts. Timings for interrupt acknowledge and return from interrupt are the same as timings for these in other Z-80 peripherals. Refer to Zilog Application Note 03-0041-01 *(The Z-80 Family Program Interrupt Structure).*

Interrupt on RDY (interrupt before requesting bus) does not directly affect the $\overline{BUSREQ}$ line. Instead, the interrupt service routine must handle this by issuing the following commands to WR6:

1. Enable after Return From Interrupt (RETI) Command — Hex B7
2. Enable DMA — Hex 87
3. An RETI instruction that resets the Interrupt Under Service latch in the Z-80 DMA.

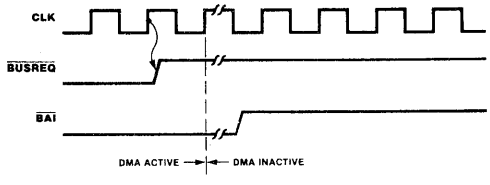


Figure 16. Bus Release (Byte-at-a-Time Mode)

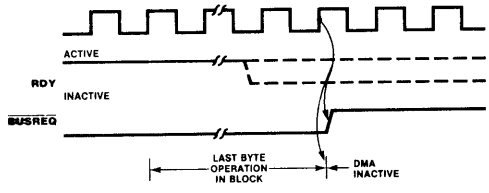


Figure 17. Bus Release at End of Block (Burst and Continuous Modes)

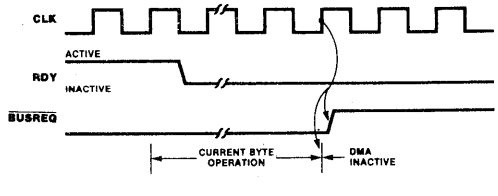


Figure 18. Bus Release When Not Ready (Burst Mode)

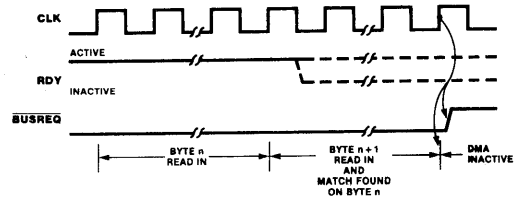


Figure 19. Bus Release on Match (Burst and Continuous Modes)

Absolute Maximum Ratings

Operating Ambient Temperature Under Bias . . . As Specified Under Ordering Information.

Storage Temperature . . . . . -65°C to +150°C

Voltage On Any Pin with Respect to Ground . . . . . -0.3 V to +7.0 V

Power Dissipation . . . . . 1.5 W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

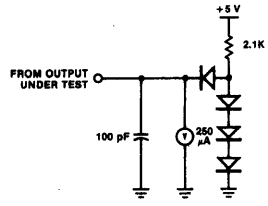
Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a load capacitance of 100 pF max. Timing references between two output signals assume a load difference of 50 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	5.5	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	5.5	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 3.2mA for $\overline{BUSREQ}$ I_{OL} = 2.0 mA for all others
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 250 µA
I_{CC}	Power Supply Current Z-80 DMA Z-80A DMA		 150 200	 mA mA	
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10	µA	V_{OUT} = 0.4 V to V_{CC}
I_{LD}	Data Bus Leakage Current in Input Mode		±10	µA	0 ≤ V_{IN} ≤ V_{CC}

V_{CC} = 5 V ±5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		35	pF	Unmeasured Pins
C_{IN}	Input Capacitance		5	pF	Returned to Ground
C_{OUT}	Output Capacitance		10	pF	

Over specified temperature range; f = 1 MHz

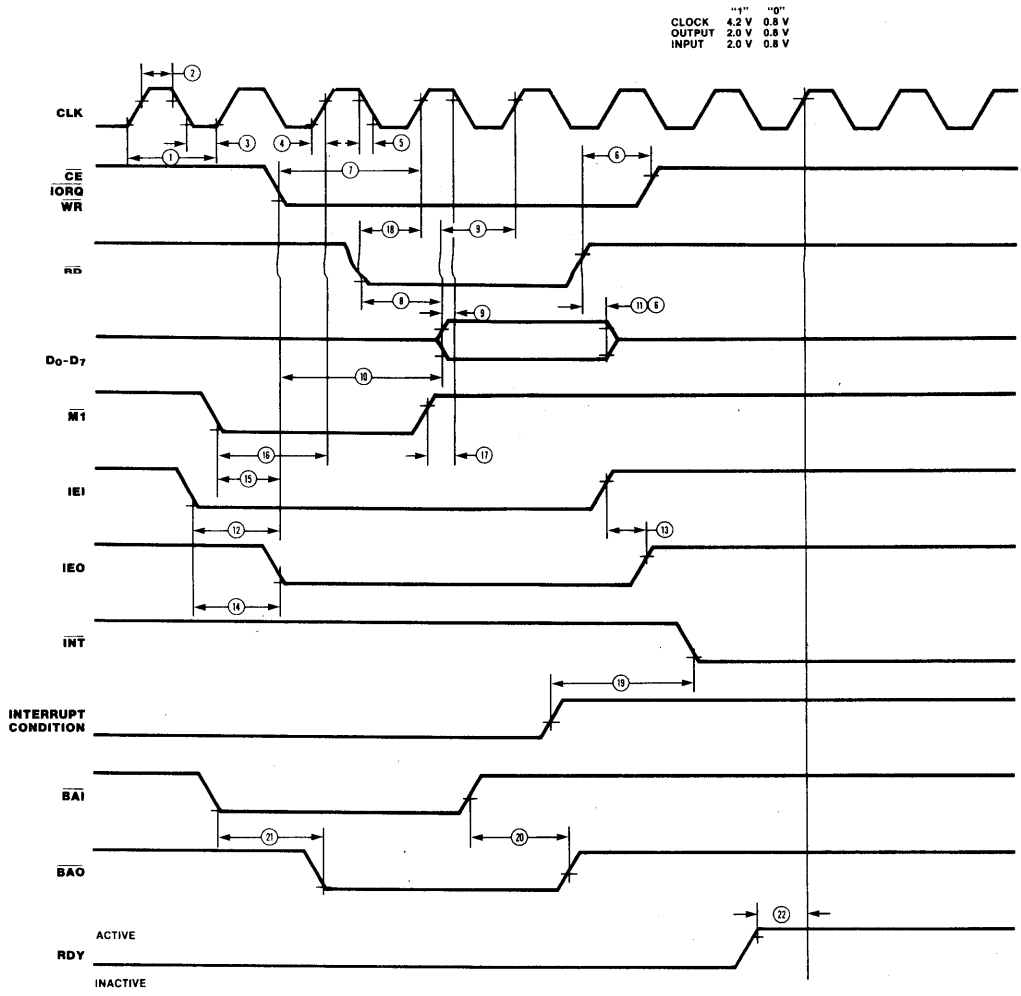
Inactive State AC Characteristics

Number	Symbol	Parameter	Z-80 DMA Min	Z-80 DMA Max	Z-80A DMA Min	Z-80A DMA Max	Unit
1	TcC	Clock Cycle Time	400	4000	250	4000	ns
2	TwCh	Clock Width (High)	170	2000	110	2000	ns
3	TwCl	Clock Width (Low)	170	2000	110	2000	ns
4	TrC	Clock Rise Time		30		30	ns
5	TfC	Clock Fall Time		30		30	ns
6	Th	Hold Time for Any Specified Setup Time	0		0		ns
7	TsC(Cr)	$\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$ ↓ to Clock ↑ Setup	280		145		ns
8	TdDO(RDf)	$\overline{RD}$↓ to Data Output Delay		500		380	ns
9	TsWM(Cr)	Data In to Clock ↑ Setup ($\overline{WR}$ or $\overline{M1}$)	50		50		ns
10	TdCf(DO)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160	ns
11	TdRD(Dz)	$\overline{RD}$ ↑ to Data Float Delay (output buffer disable)		160		110	ns
12	TsIEI(IORQ)	IEI↓ to $\overline{IORQ}$ ↓ Setup (INTA Cycle)	140		140		ns
13	TdIEOr(IEIr)	IEI ↑ to IEO ↑ Delay		210		160	ns
14	TdIEOf(IEIf)	IEI ↓ to IEO ↓ Delay		190		130	ns
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt just prior to $\overline{M1}$↓)		300		190	ns
16	TsM1f(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup	210		90		ns
17	TsM1r(Cf)	$\overline{M1}$ ↑ to Clock ↓ Setup	20		-10		ns
18	TsRD(Cr)	$\overline{RD}$ ↓ to Clock ↑ Setup ($\overline{M1}$ Cycle)	240		115		ns
19	TdI(INT)	Interrupt Cause to $\overline{INT}$ ↓ Delay ($\overline{INT}$ generated only when DMA is inactive)		500		500	ns
20	TdBAIr(BAOr)	$\overline{BAI}$ ↑ to $\overline{BAO}$ ↑ Delay		200		150	ns
21	TdBAIf(BAOf)	$\overline{BAI}$ ↓ to $\overline{BAO}$ ↓ Delay		200		150	ns
22	TsRDY(Cr)	RDY Active to Clock ↑ Setup	150		100		ns

*NOTE:

1. Negative minimum setup values mean that the first-mentioned event can come after the second-mentioned event.

Inactive State AC Characteristics (Continued)



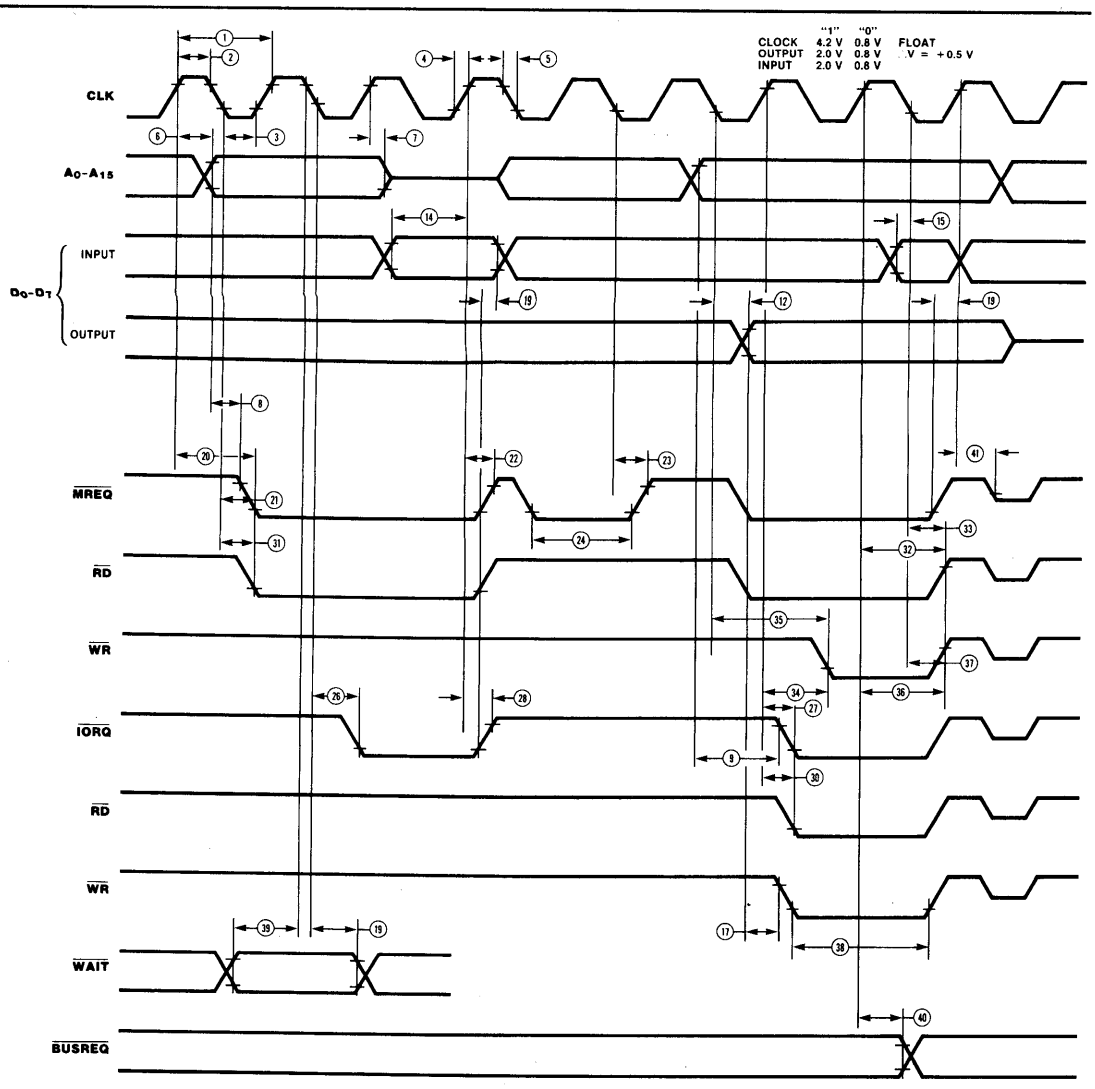
NOTE:
Signals in this diagram bear no relation to one another unless specifically noted as a numbered item.

Active State AC Characteristics

Number	Symbol	Parameter	Z-80 DMA Min(ns)	Z-80 DMA Max(ns)	Z-80A DMA Min(ns)	Z-80A DMA Max(ns)
1	TcC	Clock Cycle Time	400		250	
2	TwCh	Clock Width (High)	180	2000	110	2000
3	TwCl	Clock Width (Low)	180	2000	110	2000
4	TrC	Clock Rise Time		30		30
5	TfC	Clock Fall Time		30		30
6	TdA	Address Output Delay		145		110
7	TdC(Az)	Clock ↑ to Address Float Delay		110		90
8	TsA(MREQ)	Address to $\overline{MREQ}$ ↓ Setup (Memory Cycle)	(2)+(5)-75		(2)+(5)-75	
9	TsA(IRW)	Address Stable to $\overline{IORQ}$, $\overline{RD}$, $\overline{WR}$ ↓ Setup (I/O Cycle)	(1)-80		(1)-70	
*10	TdRW(A)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Stable Delay	(3)+(4)-40		(3)+(4)-50	
*11	TdRW(Az)	$\overline{RD}$, $\overline{WR}$ ↑ to Addr. Float	(3)+(4)-60		(3)+(4)-45	
12	TdCf(DO)	Clock ↓ to Data Out Delay		230		150
*13	TdCr(Dz)	Clock ↑ to Data Float Delay (Write Cycle)		90		90
14	TsDI(Cr)	Data In to Clock ↑ Setup (Read cycle when rising edge ends read)	50		35	
15	TsDI(Cf)	Data In to Clock ↓ Setup (Read cycle when falling edge ends read)	60		50	
*16	TsDO(WfM)	Data Out to $\overline{WR}$ ↓ Setup (Memory Cycle)	(1)-210		(1)-170	
17	TsDO(WfI)	Data Out to $\overline{WR}$ ↓Setup (I/O cycle)	100		100	
*18	TdWr(DO)	$\overline{WR}$ ↑ to Data Out Delay	(3)+(4)-80		(3)+(4)-70	
19	Th	Hold Time for Any Specified Setup Time	0		0	
20	TdCr(Mf)	Clock ↑ to $\overline{MREQ}$ ↓ Delay		100		85
21	TdCf(Mf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		100		85
22	TdCr(Mr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay		100		85
23	TdCf(Mr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay		100		85
24	TwMl	$\overline{MREQ}$ Low Pulse Width	(1)-40		(1)-30	
*25	TwMh	$\overline{MREQ}$ High Pulse Width	(2)+(5)-30		(2)+(5)-20	
26	TdCf(If)	Clock ↓ to $\overline{IORQ}$ ↓ Delay		110		85
27	TdCr(If)	Clock ↑ to $\overline{IORQ}$ ↓ Delay		90		75
28	TdCr(Ir)	Clock ↑ to $\overline{IORQ}$ ↑ Delay		100		85
*29	TdCf(Ir)	Clock ↓ to $\overline{IORQ}$ ↑ Delay		110		85
30	TdCr(Rf)	Clock ↑ to $\overline{RD}$ ↓ Delay		100		85
31	TdCf(Rf)	Clock ↓ to $\overline{RD}$ ↓ Delay		130		95
32	TdCr(Rr)	Clock ↑ to $\overline{RD}$ ↑ Delay		100		85
33	TdCf(Rr)	Clock ↓ to $\overline{RD}$ ↑ Delay		110		85
34	TdCr(Wf)	Clock ↑ to $\overline{WR}$ ↓ Delay		80		65
35	TdCf(Wf)	Clock ↓ to $\overline{WR}$ ↓ Delay		90		80
36	TdCr(Wr)	Clock ↑ to $\overline{WR}$ ↑ Delay		100		80
37	TdCf(Wr)	Clock ↓ to $\overline{WR}$ ↑ Delay		100		80
38	TwWl	$\overline{WR}$ Low Pulse Width	(1)-40		(1)-30	
39	TsWA(Cf)	$\overline{WAIT}$ to Clock ↓ Setup	70		70	
40	TdCr(B)	Clock ↑ to $\overline{BUSREQ}$ Delay		150		100
41	TdCr(Iz)	Clock ↑ to $\overline{IORQ}$, $\overline{MREQ}$, $\overline{RD}$, $\overline{WR}$ Float Delay		100		80

NOTES:
1. Numbers in parentheses are other parameter-numbers in this table; their values should be substituted in equations.
2. All equations imply DMA default (standard) timing.
3. Data must be enabled onto data bus when RD is active.
4. Asterisk (*) before parameter number means the parameter is not illustrated in the AC Timing Diagrams.

Active State AC Characteristics (Continued)



NOTE:
Signals in this diagram bear no relation to one another unless specifically noted as a numbered item.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8410	CE	2.5 MHz	Z80 DMA (40-pin)	Z8410	PE	2.5 MHz	Z80 DMA (40-pin)
Z8410	CM	2.5 MHz	Same as above	Z8410	PS	2.5 MHz	Same as above
Z8410	CS	2.5 MHz	Same as above	Z8410A	CS	4.0 MHz	Z80A DMA (40-pin)
Z8410	DE	2.5 MHz	Same as above	Z8410A	DS	4.0 MHz	Same as above
Z8410	DS	2.5 MHz	Same as above	Z8410A	PS	4.0 MHz	Same as above

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, S = 0°C to +70°C.

Z8420
Z80® PIO Parallel Input/Output Controller

Zilog

Product Specification

September 1983

Features

- Provides a direct interface between Z-80 microcomputer systems and peripheral devices.
- Both ports have interrupt-driven handshake for fast response.
- Four programmable operating modes: byte input, byte output, byte input/output (Port A only), and bit input/output.
- Programmable interrupts on peripheral status conditions.
- Standard Z-80 Family bus-request and prioritized interrupt-request daisy chains implemented without external logic.
- The eight Port B outputs can drive Darlington transistors (1.5 mA at 1.5 V).

General Description

The Z-80 PIO Parallel I/O Circuit is a programmable, dual-port device that provides a TTL-compatible interface between peripheral devices and the Z-80 CPU. The CPU configures the Z-80 PIO to interface with a wide range of peripheral devices with no other external logic. Typical peripheral devices that are compatible with the Z-80 PIO include most keyboards, paper tape readers and punches, printers, PROM programmers, etc.

One characteristic of the Z-80 peripheral controllers that separates them from other interface controllers is that all data transfer between the peripheral device and the CPU is accomplished under interrupt control. Thus, the interrupt logic of the PIO permits full use of the efficient interrupt capabilities of the Z-80 CPU during I/O transfers. All logic necessary to implement a fully nested interrupt structure is included in the PIO.

Another feature of the PIO is the ability to interrupt the CPU upon occurrence of specified status conditions in the peripheral device. For example, the PIO can be programmed to interrupt if any specified peripheral alarm conditions should occur. This interrupt capability reduces the time the processor must spend in polling peripheral status.

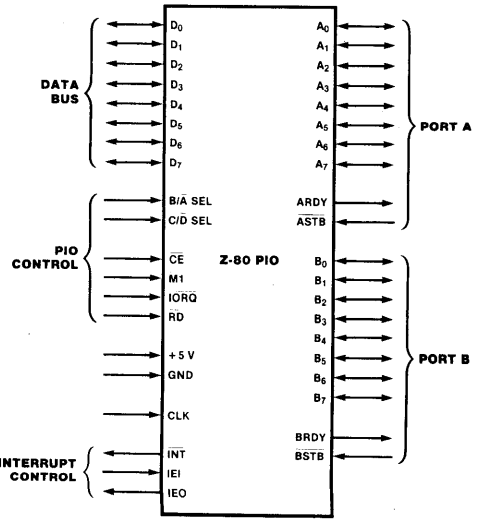


Figure 1. Pin Functions

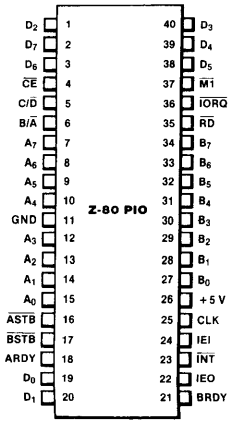


Figure 2. Pin Assignments

General Description (Continued)

The Z-80 PIO interfaces to peripherals via two independent general-purpose I/O ports, designated Port A and Port B. Each port has eight data bits and two handshake signals, Ready and Strobe, which control data transfer. The Ready output indicates to the peripheral that the port is ready for a data transfer. Strobe is an input from the peripheral that indicates when a data transfer has occurred.

Operating Modes. The Z-80 PIO ports can be programmed to operate in four modes: byte output (Mode 0), byte input (Mode 1), byte input/output (Mode 2) and bit input/output (Mode 3).

In Mode 0, either Port A or Port B can be programmed to output data. Both ports have output registers that are individually addressed by the CPU; data can be written to either port at any time. When data is written to a port, an active Ready output indicates to the external device that data is available at the associated port and is ready for transfer to the external device. After the data transfer, the external device responds with an active Strobe input, which generates an interrupt, if enabled.

In Mode 1, either Port A or Port B can be configured in the input mode. Each port has an input register addressed by the CPU. When the CPU reads data from a port, the PIO sets the Ready signal, which is detected by the external device. The external device then places data on the I/O lines and strobes the I/O port, which latches the data into the Port Input Register, resets Ready, and triggers the Interrupt Request, if enabled. The CPU can read the input data at any time, which again sets Ready.

Mode 2 is bidirectional and uses Port A, plus the interrupts and handshake signals from both ports. Port B must be set to Mode 3 and masked off. In operation, Port A is used for both data input and output. Output operation is similar to Mode 0 except that data is allowed out onto the Port A bus only when $\overline{ASTB}$ is Low. For input, operation is similar to Mode 1, except that the data input uses the Port B handshake signals and the Port B interrupt (if enabled).

Both ports can be used in Mode 3. In this mode, the individual bits are defined as either input or output bits. This provides up to eight separate, individually defined bits for each port. During operation, Ready and Strobe are not used. Instead, an interrupt is generated if the condition of one input changes, or if all inputs change. The requirements for generating an interrupt are defined during the programming operation; the active level is specified as either High or Low, and the logic condition is specified as either one input active (OR) or all inputs active (AND). For example, if the port is programmed for active Low inputs and the logic function is AND, then all inputs at the specified port must go Low to generate an interrupt.

Data outputs are controlled by the CPU and can be written or changed at any time.

- Individual bits can be masked off.
- The handshake signals are not used in Mode 3; Ready is held Low, and Strobe is disabled.
- When using the Z-80 PIO interrupts, the Z-80 CPU interrupt mode must be set to Mode 2.

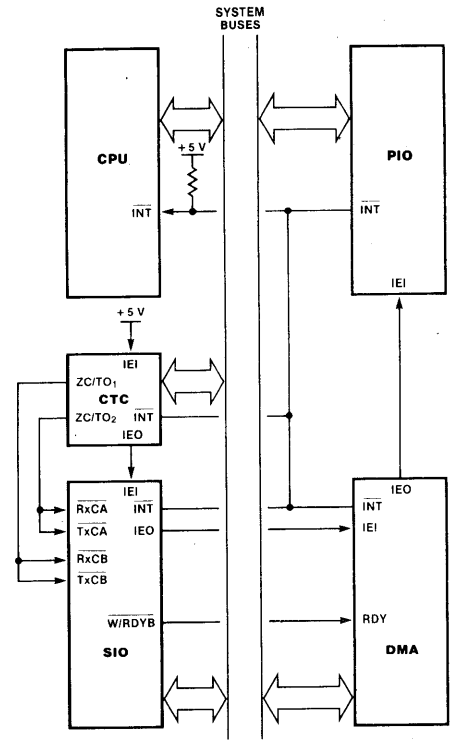


Figure 3. PIO in a Typical Z80 Family Environment

Internal Structure

The internal structure of the Z-80 PIO consists of a Z-80 CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic (Figure 4). The CPU bus interface logic allows the Z-80 PIO to interface directly to the Z-80 CPU with no other external logic. The internal control logic synchronizes the CPU data bus to the peripheral device interfaces (Port A and Port B). The two I/O ports (A and B) are virtually identical and are used to interface directly to peripheral devices.

Port Logic. Each port contains separate input and output registers, handshake control logic, and the control registers shown in Figure 5. All data transfers between the peripheral unit and the CPU use the data input and output registers. The handshake logic associated with each port controls the data transfers through the input and the output registers. The mode control register (two bits) selects one of the four programmable operating modes.

The control mode (Mode 3) uses the remaining registers. The input/output control register specifies which of the eight data bits in the port are to be outputs and enables these bits; the remaining bits are inputs. The mask register and the mask control register control Mode 3 interrupt conditions. The mask register specifies which of the bits in the port are active and which are masked or inactive.

The mask control register specifies two conditions: first, whether the active state of the input bits is High or Low, and second, whether an interrupt is generated when any *one* unmasked input bit is active (OR condition) or if the interrupt is generated when *all* unmasked input bits are active (AND condition).

Interrupt Control Logic. The interrupt control logic section handles all CPU interrupt protocol for nested-priority interrupt structures. Any device's physical location in a daisy-chain configuration determines its priority. Two lines (IEI and IEO) are provided in each PIO to form this daisy chain. The device closest to the CPU has the highest priority. Within a PIO, Port A interrupts have higher priority than those of Port B. In the byte input, byte output, or bidirectional modes, an interrupt can be generated whenever the peripheral requests a new byte transfer. In the bit control mode, an interrupt can be generated when the peripheral status matches a programmed value. The PIO provides for complete control of nested interrupts. That is, lower priority devices may not interrupt higher priority devices that have not had their interrupt service routines completed by the CPU. Higher priority devices may interrupt the servicing of lower priority devices.

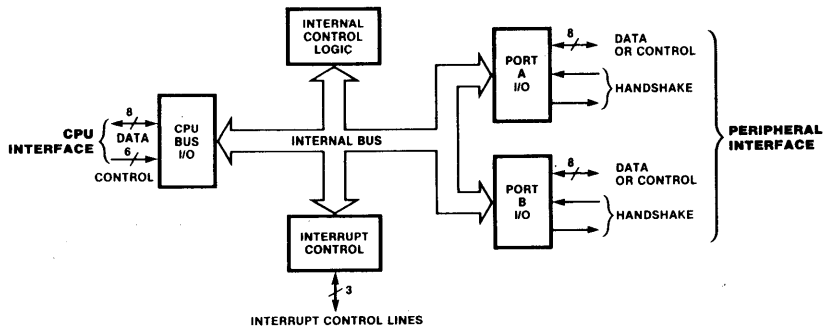


Figure 4. Block Diagram

Internal Structure (Continued)

If the CPU (in interrupt Mode 2) accepts an interrupt, the interrupting device must provide an 8-bit interrupt vector for the CPU. This vector forms a pointer to a location in memory where the address of the interrupt service routine is located. The 8-bit vector from the interrupting device forms the least significant eight bits of the indirect pointer while the I Register in the CPU provides the most significant eight bits of the pointer. Each port (A and B) has an independent interrupt vector. The least significant bit of the vector is automatically set to 0 within the PIO because the pointer must point to two adjacent memory locations for a complete 16-bit address.

Unlike the other Z-80 peripherals, the PIO does not enable interrupts immediately after programming. It waits until $\overline{M1}$ goes Low (e.g., during an opcode fetch). This condition is unimportant in the Z-80 environment but might not be if another type of CPU is used.

The PIO decodes the RETI (Return From Interrupt) instruction directly from the CPU data bus so that each PIO in the system knows at all times whether it is being serviced by the CPU interrupt service routine. No other communication with the CPU is required.

CPU Bus I/O Logic. The CPU bus interface logic interfaces the Z-80 PIO directly to the Z-80 CPU, so no external logic is necessary. For large systems, however, address decoders and/or buffers may be necessary.

Internal Control Logic. This logic receives the control words for each port during programming and, in turn, controls the operating functions of the Z-80 PIO. The control logic synchronizes the port operations, controls the port mode, port addressing, selects the read/write function, and issues appropriate commands to the ports and the interrupt logic. The Z-80 PIO does not receive a write input from the CPU; instead, the $\overline{RD}$, $\overline{CE}$, C/$\overline{D}$ and $\overline{IORQ}$ signals generate the write input internally.

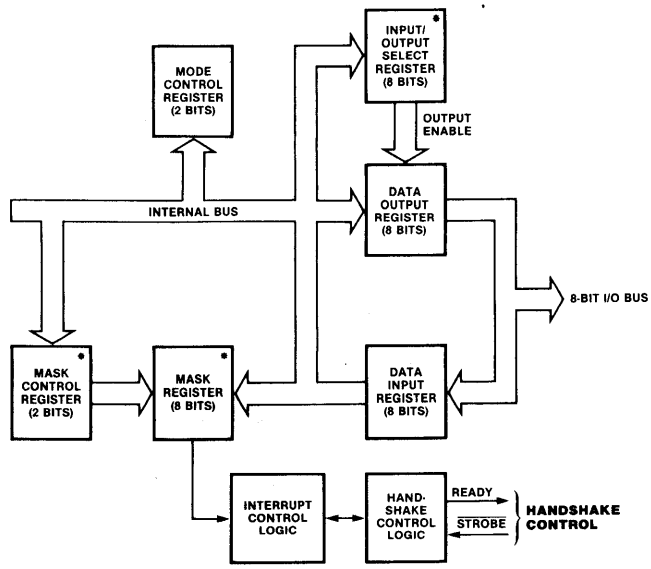


Figure 5. Typical Port I/O Block Diagram

Programming

Mode 0, 1, or 2. *(Byte Input, Output, or Bidirectional).* Programming a port for Mode 0, 1, or 2 requires two words per port. These words are:

A Mode Control Word. Selects the port operating mode (Figure 6). This word may be written any time.

An Interrupt Vector. The Z-80 PIO is designed for use with the Z-80 CPU in interrupt Mode 2 (Figure 7). When interrupts are enabled, the PIO must provide an interrupt vector.

Mode 3. (Bit Input/Output). Programming a port for Mode 3 operation requires a control word, a vector (if interrupts are enabled), and three additional words, described as follows:

I/O Register Control. When Mode 3 is selected, the mode control word must be followed by another control word that sets the I/O control register, which in turn defines which port lines are inputs and which are outputs (Figure 8).

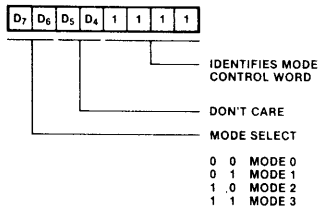


Figure 6. Mode Control Word

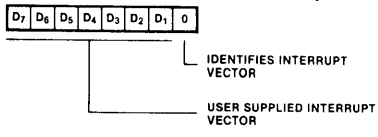


Figure 7. Interrupt Vector Word

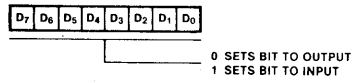


Figure 8. I/O Register Control Word

Interrupt Control Word. In Mode 3, handshake is not used. Interrupts are generated as a logic function of the input signal levels. The interrupt control word sets the logic conditions and the logic levels required for generating an interrupt. Two logic conditions or functions are available: AND (if all input bits change to the active level, an interrupt is triggered), and OR (if any one of the input bits changes to the active level, an interrupt is triggered). Bit D_6 sets the logic function, as shown in Figure 9. The active level of the input bits can be set either High or Low. The active level is controlled by Bit D_5.

Mask Control Word. This word sets the mask control register, allowing any unused bits to be masked off. If any bits are to be masked, then D_4 must be set. When D_4 is set, the next word written to the port must be a mask control word (Figure 10).

Interrupt Disable. There is one other control word which can be used to enable or disable a port interrupt. It can be used without changing the rest of the interrupt control word (Figure 11).

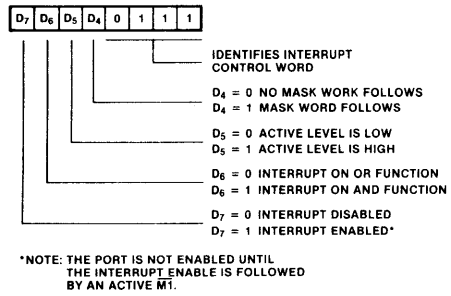


Figure 9. Interrupt Control Word

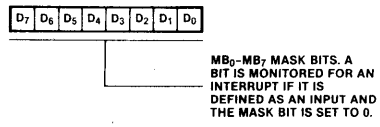


Figure 10. Mask Control Word

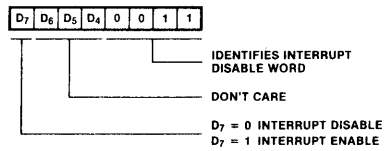


Figure 11. Interrupt Disable Word

Pin Description

A_0-A_7. *Port A Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port A of the PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

ARDY. *Register A Ready* (output, active High). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

Input Mode. This signal is active when the Port A input register is empty and ready to accept data from the peripheral device.

Bidirectional Mode. This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode, data is not placed on the Port A data bus, unless $\overline{\text{ASTB}}$ is active.

Control Mode. This signal is disabled and forced to a Low state.

$\overline{\text{ASTB}}$. *Port A Strobe Pulse From Peripheral Device* (input, active Low). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

Input Mode. The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

Bidirectional Mode. When this signal is active, data from the Port A output register is gated onto the Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

Control Mode. The strobe is inhibited internally.

B_0-B_7. *Port B Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port B and a peripheral device. The Port B data bus can supply 1.5 mA at 1.5 V to drive Darlington transistors. B_0 is the least significant bit of the bus.

B/$\overline{\text{A}}$. *Port B Or A Select* (input, High = B). This pin defines which port is accessed during a data transfer between the CPU and the PIO. A Low on this pin selects Port A; a High selects Port B. Often address bit A_0 from the CPU is used for this selection function.

BRDY. *Register B Ready* (output, active High). This signal is similar to ARDY, except that in the Port A bidirectional mode this signal is High when the Port A input register is empty and ready to accept data from the peripheral device.

$\overline{\text{BSTB}}$. *Port B Strobe Pulse From Peripheral Device* (input, active Low). This signal is similar to $\overline{\text{ASTB}}$, except that in the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

C/$\overline{\text{D}}$. *Control Or Data Select* (input, High = C). This pin defines the type of data transfer to be performed between the CPU and the PIO. A High on this pin during a CPU write to the PIO causes the Z-80 data bus to be interpreted as a *command* for the port selected by the B/$\overline{\text{A}}$ Select line. A Low on this pin means that the Z-80 data bus is being used to transfer data between the CPU and the PIO. Often address bit A_1 from the CPU is used for this function.

$\overline{\text{CE}}$. *Chip Enable* (input, active Low). A Low on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally decoded from four I/O port numbers for Ports A and B, data, and control.

CLK. *System Clock* (input). The Z-80 PIO uses the standard single-phase Z-80 system clock.

D_0-D_7. *Z-80 CPU Data Bus* (bidirectional, 3-state). This bus is used to transfer all data and commands between the Z-80 CPU and the Z-80 PIO. D_0 is the least significant bit.

IEI. *Interrupt Enable In* (input, active High). This signal is used to form a priority-interrupt daisy chain when more than one interrupt-driven device is being used. A High level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). The IEO signal is the other signal required to form a daisy chain priority scheme. It is High only if IEI is High and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{\text{INT}}$. *Interrupt Request* (output, open drain, active Low). When $\overline{\text{INT}}$ is active the Z-80 PIO is requesting an interrupt from the Z-80 CPU.

$\overline{\text{IORQ}}$. *Input/Output Request* (input from Z-80 CPU, active Low). $\overline{\text{IORQ}}$ is used in conjunction with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$, and $\overline{\text{RD}}$ to transfer commands and data between the Z-80 CPU and the Z-80 PIO. When $\overline{\text{CE}}$, $\overline{\text{RD}}$, and $\overline{\text{IORQ}}$ are active, the port addressed by B/$\overline{\text{A}}$ transfers data to the CPU (a read operation). Conversely, when $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ are active but $\overline{\text{RD}}$ is not, the port addressed by B/$\overline{\text{A}}$ is written into from the CPU with either data or control information, as specified by C/$\overline{\text{D}}$. Also, if $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt; the interrupting port automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

Pin Description (Continued)

$\overline{M1}$. *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{M1}$ and $\overline{RD}$ signals are active, the Z-80 CPU is fetching an instruction from memory. Conversely, when both $\overline{M1}$ and $\overline{IORQ}$ are active, the CPU is acknowledging an interrupt. In addition, $\overline{M1}$ has two other functions within the Z-80 PIO: it synchronizes the PIO interrupt logic; when $\overline{M1}$ occurs without an active $\overline{RD}$ or $\overline{IORQ}$ signal, the PIO is reset.

$\overline{RD}$. *Read Cycle Status* (input from Z-80 CPU, active Low). If $\overline{RD}$ is active, or an I/O operation is in progress, $\overline{RD}$ is used with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$, and $\overline{IORQ}$ to transfer data from the Z-80 PIO to the Z-80 CPU.

Timing

The following timing diagrams show typical timing in a Z-80 CPU environment. For more precise specifications refer to the composite ac timing diagram.

Write Cycle. Figure 12 illustrates the timing for programming the Z-80 PIO or for writing data to one of its ports. No Wait states are allowed for writing to the PIO other than the automatically inserted T_{WA}. The PIO does not receive a specific write signal; it internally generates its own from the lack of an active $\overline{RD}$ signal.

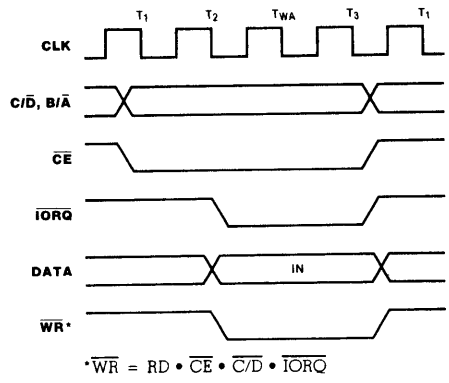


Figure 12. Write Cycle Timing

Read Cycle. Figure 13 illustrates the timing for reading the data input from an external device to one of the Z-80 PIO ports. No Wait states are allowed for reading the PIO other than the automatically inserted T_{WA}.

Output Mode (Mode 0). An output cycle (Figure 14) is always started by the execution of an output instruction by the CPU. The $\overline{WR}$* pulse from the CPU latches the data from the CPU data bus into the selected port's output register. The $\overline{WR}$* pulse sets the Ready flag after a Low-going edge of CLK, indicating data is available. Ready stays active until the positive edge of the strobe line is received, indicating that data was taken by the peripheral. The positive edge of the strobe pulse generates an $\overline{INT}$ if the interrupt enable flip-flop has been set and if this device has the highest priority.

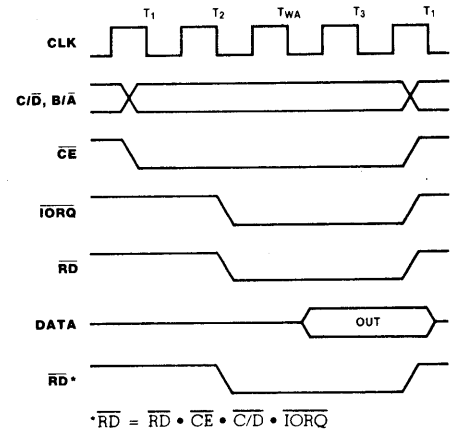


Figure 13. Read Cycle Timing

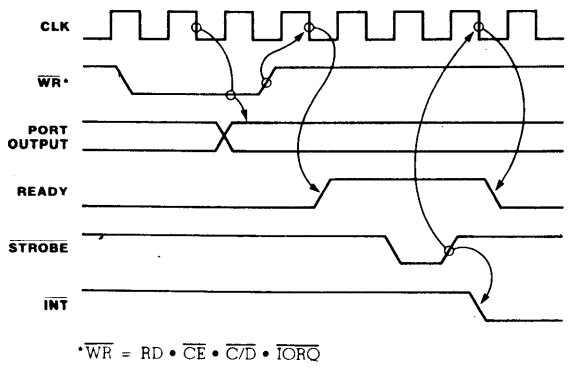


Figure 14. Mode 0 Output Timing

Timing (Continued)

Input Mode (Mode 1). When $\overline{\text{STROBE}}$ goes Low, data is loaded into the selected port input register (Figure 15). The next rising edge of strobe activates $\overline{\text{INT}}$, if Interrupt Enable is set and this is the highest-priority requesting device. The following falling edge of CLK resets Ready to an inactive state, indicating that the input register is full and cannot accept any more data until the CPU completes a read. When a read is complete, the positive edge of $\overline{\text{RD}}$ sets Ready at the next Low-going transition of CLK. At this time new data can be loaded into the PIO.

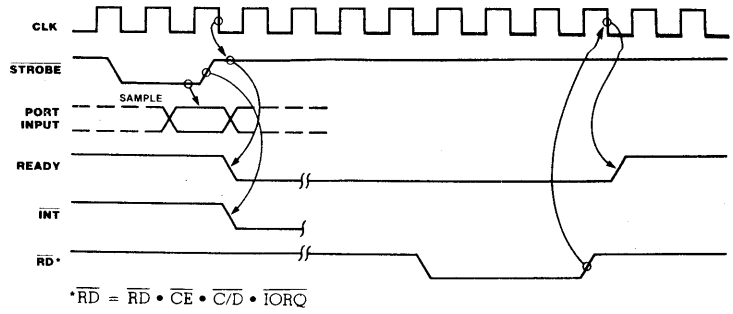


Figure 15. Mode 1 Input Timing

Bidirectional Mode (Mode 2). This is a combination of Modes 0 and 1 using all four handshake lines and the eight Port A I/O lines (Figure 16). Port B must be set to the bit mode and its inputs must be masked. The Port A handshake lines are used for output control and the Port B lines are used for input control. If interrupts occur, Port A's vector will be used during port output and Port B's will be used during port input. Data is allowed out onto the Port A bus only when $\overline{\text{ASTB}}$ is Low. The rising edge of this strobe can be used to latch the data into the peripheral.

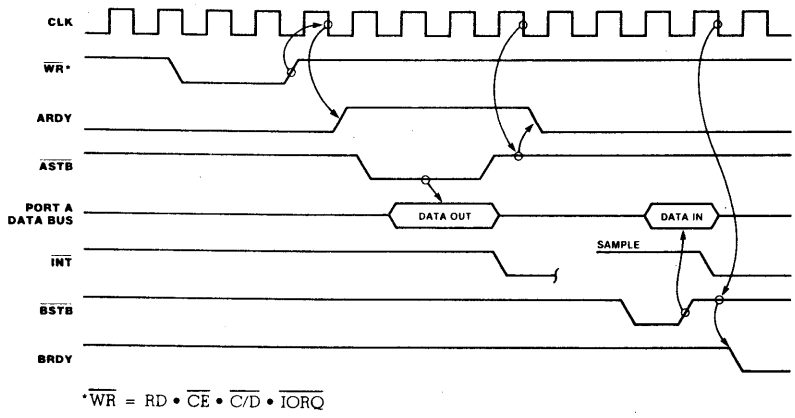
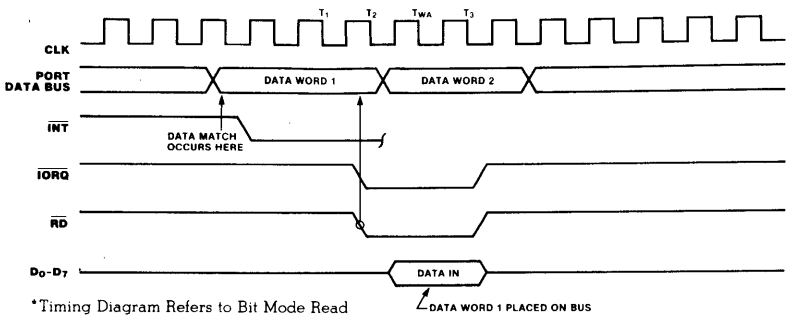


Figure 16. Mode 2 Bidirectional Timing

Timing (Continued)

Bit Mode (Mode 3). The bit mode does not utilize the handshake signals, and a normal port write or port read can be executed at any time. When writing, the data is latched into the output registers with the same timing as the output mode (Figure 17).

When reading the PIO, the data returned to the CPU is composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register contains data that was present immediately prior to the falling edge of $\overline{RD}$. An interrupt is generated if interrupts from the port are enabled and the data on the port data lines satisfy the logical equation defined by the 8-bit mask and 2-bit mask control registers. However, if Port A is programmed in bidirectional mode, Port B does not issue an interrupt in bit mode and must therefore be polled.



*Timing Diagram Refers to Bit Mode Read

Figure 17. Mode 3 Bit Mode Timing

Interrupt Acknowledge Timing. During $\overline{M1}$ time, peripheral controllers are inhibited from changing their interrupt enable status, permitting the Interrupt Enable signal to ripple through the daisy chain. The peripheral with IEI High and IEO Low during $\overline{INTACK}$ places a preprogrammed 8-bit interrupt vector on the data bus at this time (Figure 18). IEO is held Low until a Return From Interrupt (RETI) instruction is executed by the CPU while IEI is High. The 2-byte RETI instruction is decoded internally by the PIO for this purpose.

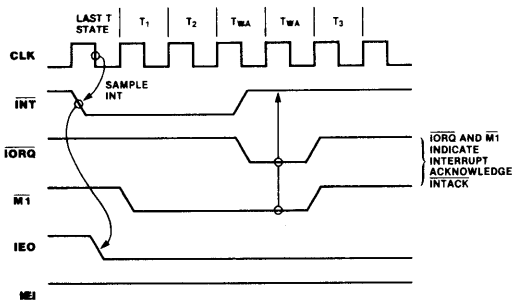


Figure 18. Interrupt Acknowledge Timing

Return From Interrupt Cycle. If a Z-80 peripheral has no interrupt pending and is not under service, then its IEO = IEI. If it has an interrupt under service (i.e., it has already interrupted and received an interrupt acknowledge) then its IEO is always Low, inhibiting lower priority devices from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO is Low unless an "ED" is decoded as the first byte of a 2-byte opcode (Figure 19). In this case, IEO goes High until the next opcode byte is decoded, whereupon it goes Low again. If the second byte of the opcode was a "4D," then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service has its IEI High and its IEO Low. This device is the highest-priority device in the daisy chain that has received an interrupt acknowledge. All other peripherals have IEI = IEO. If the next opcode byte decoded is "4D," this peripheral device resets its "interrupt under service" condition.

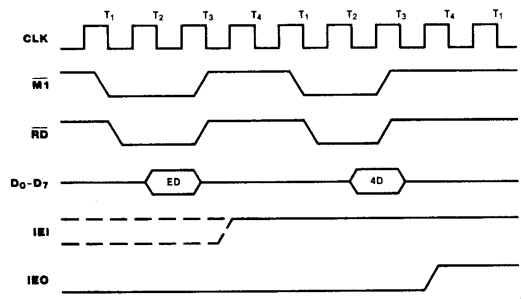
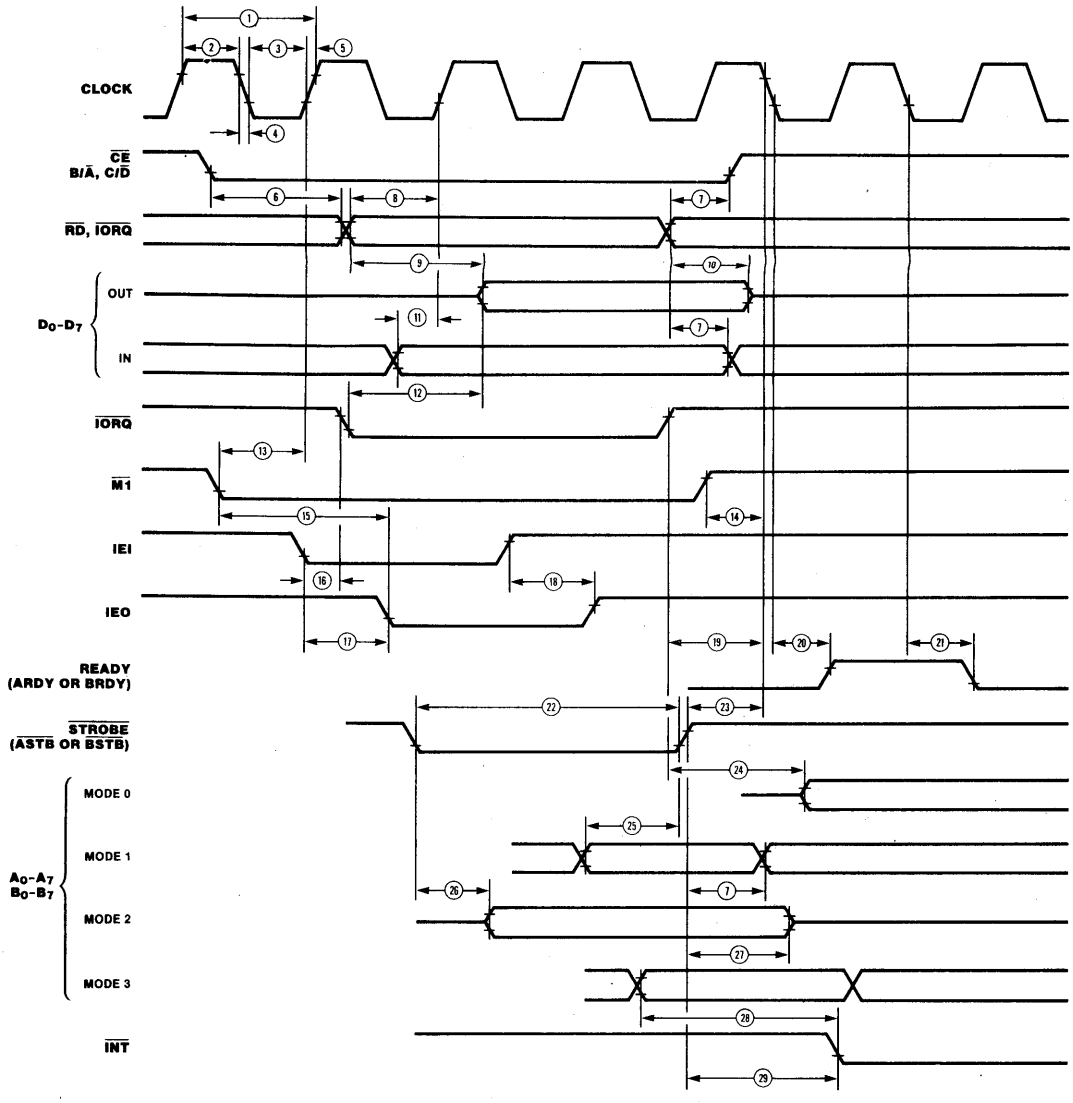


Figure 19. Return From Interrupt

AC Characteristics



Number	Symbol	Parameter	Z-80 PIO Min (ns)	Z-80 PIO Max (ns)	Z-80A PIO Min (ns)	Z-80A PIO Max (ns)	Z-80B PIO[9] Min (ns)	Z-80B PIO[9] Max (ns)	Comment
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCh	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	TsCS(RI)	$\overline{CE}$, B/$\overline{A}$, C/$\overline{D}$ to $\overline{RD}$, $\overline{IORQ}$ ↓ Setup Time	50		50		50		[6]
7	Th	Any Hold Times for Specified Setup Time	0		0		0	0	
8	TsRI(C)	$\overline{RD}$, $\overline{IORQ}$ to Clock ↑ Setup Time	115		115		70		
9	TdRI(DO)	$\overline{RD}$, $\overline{IORQ}$ ↓to Data Out Delay		430		380		300	[2]
10	TdRI(DOs)	$\overline{RD}$, $\overline{IORQ}$ ↑ to Data Out Float Delay		160		110		70	
11	TsDI(C)	Data In to Clock ↑ Setup Time	50		50		40		CL = 50 pF
12	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		120	[3]
13	TsM1(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup Time	210		90		70		
14	TsM1(Cf)	$\overline{M1}$ ↑ to Clock ↓ Setup Time ($\overline{M1}$ Cycle)	0		0		0		[8]
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt Immediately Preceding $\overline{M1}$ ↓)		300		190		100	[5, 7]
16	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	140		140		100		[7]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		120	[5] CL = 50 pF
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED Decode)		210		160		160	[5]
19	TcIO(C)	$\overline{IORQ}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		200		170		
20	TdC(RDYr)	Clock ↓ to READY ↑ Delay		200		190		170	[5] CL = 50 pF
21	TdC(RDYf)	Clock ↓ to READY ↓ Delay		150		140		120	[5]
22	TwSTB	$\overline{STROBE}$ Pulse Width	150		150		120		[4]
23	TsSTB(C)	$\overline{STROBE}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	220		220		150		[5]
24	TdIO(PD)	$\overline{IORQ}$ ↑ to PORT DATA Stable Delay (Mode 0)		200		180		160	[5]
25	TsPD(STB)	PORT DATA to $\overline{STROBE}$ ↑ Setup Time (Mode 1)	260		230		190		
26	TdSTB(PD)	$\overline{STROBE}$ ↓ to PORT DATA Stable (Mode 2)		230		210		180	[5]
27	TdSTB(PDr)	$\overline{STROBE}$ ↑ to PORT DATA Float Delay (Mode 2)		200		180		160	CL = 50 pF
28	TdPD(INT)	PORT DATA Match to $\overline{INT}$ ↓ Delay (Mode 3)		540		490		430	
29	TdSTB(INT)	$\overline{STROBE}$ ↑ to $\overline{INT}$ ↓ Delay		490		440		350	

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase TdRI(DO) by 10 ns for each 50 pF increase in load up to 200 pF max.
[3] Increase TdIO(DOI) by 10 ns for each 50 pF, increase in loading up to 200 pF max.
[4] For Mode 2: TwSTB > TsPD(STB).
[5] Increase these values by 2 ns for each 10 pF increase in loading up to 100 pF max.
[6] TsCS(RI) may be reduced. However, the time subtracted from TsCS(RI) will be added to TdRI(DO).
[7] 2.5 TcC > (N-2)TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL Buffer Delay, if any.
[8] $\overline{M1}$ must be active for a minimum of two clock cycles to reset the PIO.
[9] Z80B PIO numbers are preliminary and subject to change.

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

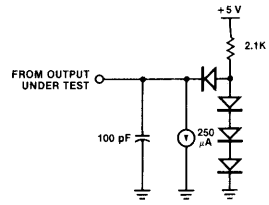
Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a load capacitance of 100 pF max.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current		±10.0	µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10.0	µA	V_{OUT} = 0.4 V to V_{CC}
I_{CC}	Power Supply Current		100.0	mA	V_{OH} = 1.5 V
I_{OHD}	Darlington Drive Current	-1.5		mA	R_{EXT} = 390 Ω

Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		10	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

Over specified temperature range; f = 1MH_z

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8420	CE	2.5 MHz	Z80 PIO (40-pin)
Z8420	CM	2.5 MHz	Same as above
Z8420	CMB	2.5 MHz	Same as above
Z8420	CS	2.5 MHz	Same as above
Z8420	DE	2.5 MHz	Same as above
Z8420	DS	2.5 MHz	Same as above
Z8420	PE	4.0 MHz	Same as above
Z8420	PS	4.0 MHz	Same as above
Z8420A	CE	4.0 MHz	Z80A PIO (40-pin)
Z8420A	CM	4.0 MHz	Same as above
Z8420A	CMB	4.0 MHz	Z80A PIO (40-pin)
Z8420A	CS	4.0 MHz	Same as above
Z8420A	DE	4.0 MHz	Same as above
Z8420A	DS	4.0 MHz	Same as above
Z8420A	PE	4.0 MHz	Same as above
Z8420A	PS	4.0 MHz	Same as above
Z8420B	CS	6,0 MHz	Same as above
Z8420B	DS	6.0 MHz	Same as above
Z8420B	PS	6.0 MHz	Same as above

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = 55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8430
Z80® CTC Counter/ Timer Circuit

Zilog

Product Specification

September 1983

Features

- Four independently programmable counter/timer channels, each with a readable downcounter and a selectable 16 or 256 prescaler. Downcounters are reloaded automatically at zero count.
- Three channels have Zero Count/Timeout outputs capable of driving Darlington transistors.
- Selectable positive or negative trigger initiates timer operation.
- Standard Z-80 Family daisy-chain interrupt structure provides fully vectored, prioritized interrupts without external logic. The CTC may also be used as an interrupt controller.
- Interfaces directly to the Z-80 CPU or—for baud rate generation—to the Z-80 SIO.

General Description

The Z-80 CTC four-channel counter/timer can be programmed by system software for a broad range of counting and timing applications. The four independently programmable channels of the Z-80 CTC satisfy common microcomputer system requirements for event counting, interrupt and interval timing, and general clock rate generation.

System design is simplified because the CTC connects directly to both the Z-80 CPU and the Z-80 SIO with no additional logic. In larger systems, address decoders and buffers may be required.

Programming the CTC is straightforward: each channel is programmed with two bytes; a third is necessary when interrupts are enabled. Once started, the CTC counts down, reloads its time constant automatically, and resumes counting. Software timing loops are completely eliminated. Interrupt processing is simplified because only one vector need be specified; the CTC internally generates a unique vector for each channel.

The Z-80 CTC requires a single +5 V power supply and the standard Z-80 single-phase system clock. It is fabricated with n-channel silicon-gate depletion-load technology, and packaged in a 28-pin plastic or ceramic DIP.

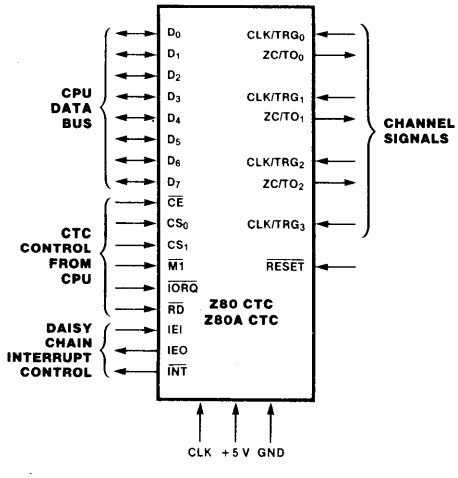


Figure 1. Pin Functions

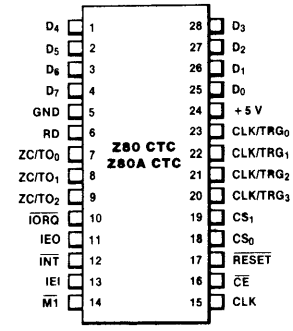


Figure 2. Pin Assignments

Functional Description

The Z-80 CTC has four independent counter/timer channels. Each channel is individually programmed with two words: a control word and a time-constant word. The control word selects the operating mode (counter or timer), enables or disables the channel interrupt, and selects certain other operating parameters. If the timing mode is selected, the control word also sets a prescaler, which divides the system clock by either 16 or 256. The time-constant word is a value from 1 to 256.

During operation, the individual counter channel counts down from the preset time constant value. In counter mode operation the counter decrements on each of the CLK/TRG input pulses until zero count is reached. Each decrement is synchronized by the system clock. For counts greater than 256, more than one counter can be cascaded. At zero count, the down-counter is automatically reset with the time constant value.

The timer mode determines time intervals as small as 4 μs (Z-80A) or 6.4 μs (Z-80) without additional logic or software timing loops. Time intervals are generated by dividing the system clock with a prescaler that decrements a preset down-counter.

Thus, the time interval is an integral multiple of the clock period, the prescaler value (16 or 256) and the time constant that is preset in the down-counter. A timer is triggered automatically when its time constant value is programmed, or by an external CLK/TRG input.

Three channels have two outputs that occur at zero count. The first output is a zero-count/timeout pulse at the ZC/TO output. The fourth channel (Channel 3) does not have a ZC/TO output; interrupt request is the only output available from Channel 3.

The second output is Interrupt Request ($\overline{INT}$), which occurs if the channel has its interrupt enabled during programming. When the Z-80 CPU acknowledges Interrupt Request, the Z-80 CTC places an interrupt vector on the data bus.

The four channels of the Z-80 CTC are fully prioritized and fit into four contiguous slots in a standard Z-80 daisy-chain interrupt structure. Channel 0 is the highest priority and Channel 3 the lowest. Interrupts can be individually enabled (or disabled) for each of the four channels.

Architecture

The CTC has four major elements, as shown in Figure 3.

- CPU bus I/O
- Channel control logic
- Interrupt logic
- Counter/timer circuits

CPU Bus I/O. The CPU bus I/O circuit decodes the address inputs, and interfaces the CPU data and control signals to the CTC for distribution on the internal bus.

Internal Control Logic. The CTC internal control logic controls overall chip operating functions such as the chip enable, reset, and read/write logic.

Interrupt Logic. The interrupt control logic ensures that the CTC interrupts interface properly with the Z-80 CPU interrupt system. The logic controls the interrupt priority of the CTC as a function of the IEI signal. If IEI is High, the CTC has priority. During interrupt

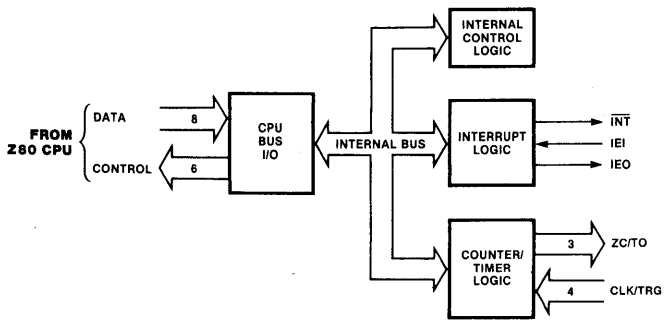


Figure 3. Functional Block Diagram

Architecture (Continued)

processing, the interrupt logic holds IEO Low, which inhibits the interrupt operation on lower priority devices. If the IEI input goes Low, priority is relinquished and the interrupt logic drives IEO Low.

If a channel is programmed to request an interrupt, the interrupt logic drives IEO Low at the zero count, and generates an $\overline{INT}$ signal to the Z-80 CPU. When the Z-80 CPU responds with interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$), then the interrupt logic arbitrates the CTC internal priorities, and the interrupt control logic places a unique interrupt vector on the data bus.

If an interrupt is pending, the interrupt logic holds IEO Low. When the Z-80 CPU issues a Return From Interrupt (RETI) instruction, each peripheral device decodes the first byte (ED_{16}). If the device has a pending interrupt, it raises IEO (High) for one $\overline{M1}$ cycle. This ensures that all lower priority devices can decode the entire RETI instruction and reset properly.

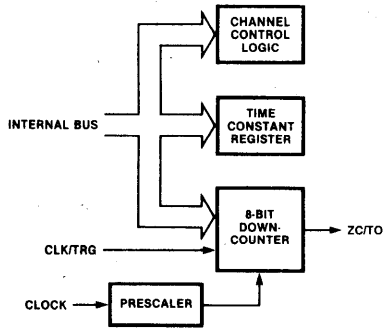


Figure 4. Counter/Timer Block Diagram

Counter/Timer Circuits. The CTC has four independent counter/timer circuits, each containing the logic shown in Figure 4.

Channel Control Logic. The channel control logic receives the 8-bit channel control word when the counter/timer channel is programmed. The channel control logic decodes the control word and sets the following operating conditions:

- Interrupt enable (or disable)
- Operating mode (timer or counter)
- Timer mode prescaler factor (16 or 256)
- Active slope for CLK/TRG input
- Timer mode trigger (automatic or CLK/TRG input)
- Time constant data word to follow
- Software reset

Time Constant Register. When the counter/timer channel is programmed, the time constant register receives and stores an 8-bit time constant value, which can be anywhere from 1 to 256 (0 = 256). This constant is automatically loaded into the down-counter when the counter/timer channel is initialized, and subsequently after each zero count.

Prescaler. The prescaler, which is used only in timer mode, divides the system clock frequency by a factor of either 16 or 256. The prescaler output clocks the down-counter during timer operation. The effect of the prescaler on the down-counter is a multiplication of the system clock period by 16 or 256. The prescaler factor is programmed by bit 5 of the channel control word.

Down-Counter. Prior to each count cycle, the down-counter is loaded with the time constant register contents. The counter is then decremented one of two ways, depending on operating mode:

- By the prescaler output (timer mode)
- By the trigger pulses into the CLK/TRG input (counter mode)

Without disturbing the down-count, the Z-80 CPU can read the count remaining at any time by performing an I/O read operation at the port address assigned to the CTC channel. When the down-counter reaches the zero count, the ZC/TO output generates a positive-going pulse. When the interrupt is enabled, zero count also triggers an interrupt request signal ($\overline{INT}$) from the interrupt logic.

Programming Each Z-80 CTC channel must be programmed prior to operation. Programming consists of writing two words to the I/O port that corresponds to the desired channel. The first word is a control word that selects the operating mode and other parameters; the second word is a time constant, which is a binary data word with a value from 1 to 256. A time constant word must be preceded by a channel control word.

After initialization, channels may be reprogrammed at any time. If updated control and time constant words are written to a channel during the count operation, the count continues to zero before the new time constant is loaded into the counter.

If the interrupt on any Z-80 CTC channel is enabled, the programming procedure should also include an interrupt vector. Only one vector is required for all four channels, because the interrupt logic automatically modifies the vector for the channel requesting service.

A control word is identified by a 1 in bit 0. A 1 in bit 2 indicates a time constant word is to follow. Interrupt vectors are always addressed to Channel 0, and identified by a 0 in bit 0.

Addressing. During programming, channels are addressed with the channel select pins CS_1 and CS_2. A 2-bit binary code selects the appropriate channel as shown in the following table.

Channel	CS_1	CS_0
0	0	0
1	0	1
2	1	0
3	1	1

Reset. The CTC has both hardware and software resets. The hardware reset terminates all down-counts and disables all CTC interrupts by resetting the interrupt bits in the control registers. In addition, the ZC/TO and Interrupt outputs go inactive, IEO reflects IEI, and D_0–D_7 go to the high-impedance state. All channels must be completely reprogrammed after a hardware reset.

The software reset is controlled by bit 1 in the channel control word. When a channel receives a software reset, it stops counting. When a software reset is used, the other bits in the control word also change the contents of the channel control register. After a software reset a new time constant word must be written *to the same channel.*

If the channel control word has both bits D_1 and D_2 set to 1, the addressed channel stops operating, pending a new time constant word. The channel is ready to resume after the new constant is programmed. In timer mode, if $D_3 = 0$, operation is triggered automatically when the time constant word is loaded.

Channel Control Word Programming. The channel control word is shown in Figure 5. It sets the modes and parameters described below.

Interrupt Enable. D_7 enables the interrupt, so that an interrupt output ($\overline{INT}$) is generated at zero count. Interrupts may be programmed in either mode and may be enabled or disabled at any time.

Operating Mode. D_6 selects either timer or counter mode.

Prescaler Factor. (Timer Mode Only). D_5 selects factor—either 16 or 256.

Trigger Slope. D_4 selects the active edge or slope of the CLK/TRG input pulses. Note that reprogramming the CLK/TRG slope during operation is equivalent to issuing an active edge. If the trigger slope is changed by a control word update while a channel is pending operation in timer mode, the result is the same as a CLK/TRG pulse and the timer starts. Similarly, if the channel is in counter mode, the counter decrements.

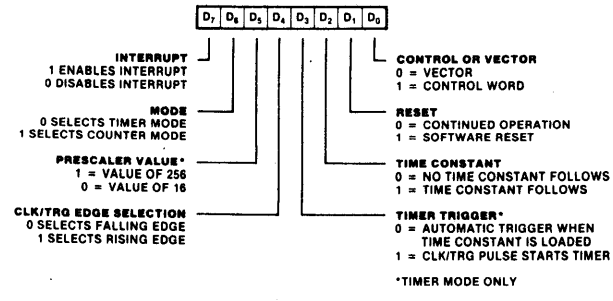


Figure 5. Channel Control Word

Programming (Continued)

Trigger Mode (Timer Mode Only). D_3 selects the trigger mode for timer operation. When D_3 is reset to 0, the timer is triggered automatically. The time constant word is programmed during an I/O write operation, which takes one machine cycle. At the end of the write operation there is a setup delay of one clock period. The timer starts automatically (decrements) on the rising edge of the second clock pulse (T_2) of the machine cycle following the write operation. Once started, the timer runs continuously. At zero count the timer reloads automatically and continues counting without interruption or delay, until stopped by a reset.

When D_3 is set to 1, the timer is triggered externally through the CLK/TRG input. The time constant word is programmed during an I/O write operation, which takes one machine cycle. The timer is ready for operation on the rising edge of the second clock pulse (T_2) of the following machine cycle. Note that the first timer decrement follows the active edge of the CLK/TRG pulse by a delay time of one clock cycle if a minimum setup time to the rising edge of clock is met. If this minimum is not met, the delay is extended by another clock period. Consequently, for immediate triggering, the CLK/TRG input must precede T_2 by one clock cycle plus its minimum setup time. If the minimum time is not met, the timer will start on the third clock cycle (T_3).

Once started the timer operates continuously, without interruption or delay, until stopped by a reset.

Time Constant to Follow. A 1 in D_2 indicates that the next word addressed to the selected channel is a time constant data word for the time constant register. The time constant word may be written at any time.

A 0 in D_2 indicates no time constant word is to follow. This is ordinarily used when the channel is already in operation and the new channel control word is an update. A channel will not operate without a time constant value. The only way to write a time constant value is to write a control word with D_2 set.

Software Reset. Setting D_1 to 1 causes a software reset, which is described in the Reset section.

Control Word. Setting D_0 to 1 identifies the word as a control word.

Time Constant Programming. Before a channel can start counting it must receive a time constant word from the CPU. During programming or reprogramming, a channel control word in which bit 2 is set must precede the time constant word to indicate that the next word is a time constant. The time constant word can be any value from 1 to 256 (Figure 6). Note that 00_{16} is interpreted as 256.

In timer mode, the time interval is controlled by three factors:

- The system clock period (ϕ)
- The prescaler factor (P), which multiplies the interval by either 16 or 256
- The time constant (T), which is programmed into the time constant register

Consequently, the time interval is the product of $\phi \times P \times T$. The minimum timer resolution is $16 \times \phi$ (4 μs with a 4 MHz clock). The maximum timer interval is $256 \times \phi \times 256$ (16.4 ms with a 4 MHz clock). For longer intervals timers may be cascaded.

Interrupt Vector Programming. If the Z-80 CTC has one or more interrupts enabled, it can supply interrupt vectors to the Z-80 CPU. To do so, the Z-80 CTC must be pre-programmed with the most-significant five bits of the interrupt vector. Programming consists of writing a vector word to the I/O port corresponding to the Z-80 CTC Channel 0. Note that D_0 of the vector word is always zero, to distinguish the vector from a channel control word. D_1 and D_2 are not used in programming the vector word. These bits are supplied by the interrupt logic to identify the channel requesting interrupt service with a unique interrupt vector (Figure 7). Channel 0 has the highest priority.

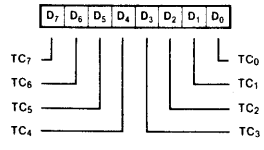


Figure 6. Time Constant Word

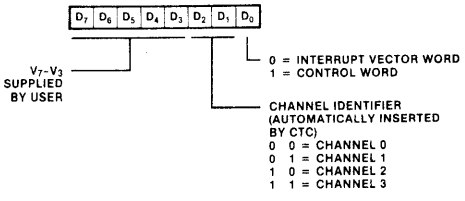


Figure 7. Interrupt Vector Word

Pin Description

$\overline{CE}$. *Chip Enable* (input, active Low). When enabled the CTC accepts control words, interrupt vectors, or time constant data words from the data bus during an I/O write cycle; or transmits the contents of the down-counter to the CPU during an I/O read cycle. In most applications this signal is decoded from the eight least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four counter-timer channels.

CLK. *System Clock* (input). *Standard single-*phase Z-80 system clock.

CLK/TRG$_0$-CLK/TRG$_3$. *External Clock/Timer Trigger* (input, user-selectable active High or Low). Four pins corresponding to the four Z-80 CTC channels. In counter mode, every active edge on this pin decrements the down-counter. In timer mode, an active edge starts the timer.

CS$_0$-CS$_1$. *Channel Select* (inputs active High). Two-bit binary address code selects one of the four CTC channels for an I/O write or read (usually connected to A_0 and A_1).

D$_0$-D$_7$. *System Data Bus* (bidirectional, 3-state). Transfers all data and commands between the Z-80 CPU and the Z-80 CTC.

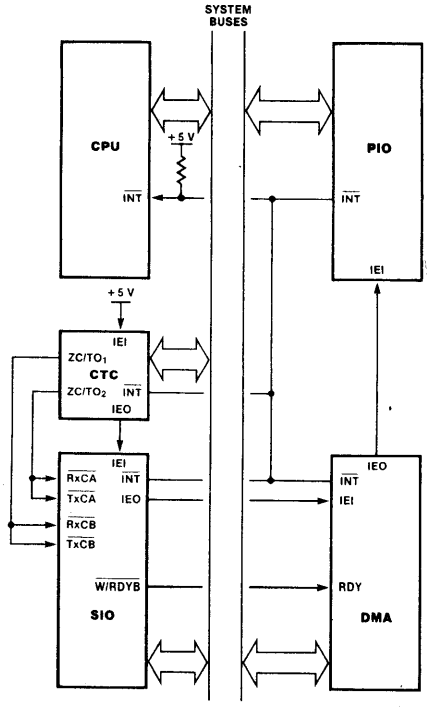


Figure 8. A Typical Z-80 Environment

$\overline{IEI}$. *Interrupt Enable In* (input, active High). A High indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z-80 CPU.

$\overline{IEO}$. *Interrupt Enable Out* (output, active High). High only if $\overline{IEI}$ is High and the Z-80 CPU is not servicing an interrupt from any Z-80 CTC channel. $\overline{IEO}$ blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). Low when any Z-80 CTC channel that has been programmed to enable interrupts has a zero-count condition in its down-counter.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). Used with $\overline{CE}$ and $\overline{RD}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC. During a write cycle, $\overline{IORQ}$ and $\overline{CE}$ are active and $\overline{RD}$ inactive. The Z-80 CTC does not receive a specific write signal; rather, it internally generates its own from the inverse of an active $\overline{RD}$ signal. In a read cycle, $\overline{IORQ}$, $\overline{CE}$ and $\overline{RD}$ are active; the contents of the down-counter are read by the Z-80 CPU. If $\overline{IORQ}$ and $\overline{M1}$ are both true, the CPU is acknowledging an interrupt request, and the highest priority interrupting channel places its interrupt vector on the Z-80 data bus.

$\overline{M1}$. *Machine Cycle One* (input from CPU, active Low). When $\overline{M1}$ and $\overline{IORQ}$ are active, the Z-80 CPU is acknowledging an interrupt. The Z-80 CTC then places an interrupt vector on the data bus if it has highest priority, and if a channel has requested an interrupt ($\overline{INT}$).

$\overline{RD}$. *Read Cycle Status* (input, active Low). Used in conjunction with $\overline{IORQ}$ and $\overline{CE}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC.

$\overline{RESET}$. *Reset* (input active Low). Terminates all down-counts and disables all interrupts by resetting the interrupt bits in all control registers; the ZC/TO and the Interrupt outputs go inactive; $\overline{IEO}$ reflects $\overline{IEI}$; D_0-D_7 go to the high-impedance state.

ZC/TO$_0$-ZC/TO$_2$. *Zero Count/Timeout* (output, active High). Three ZC/TO pins corresponding to Z-80 CTC channels 2 through 0 (Channel 3 has no ZC/TO pin). In both counter and timer modes the output is an active High pulse when the down-counter decrements to zero.

Timing

Read Cycle Timing. Figure 9 shows read cycle timing. This cycle reads the contents of a down-counter without disturbing the count. During clock cycle T_2, the Z-80 CPU initiates a read cycle by driving the following inputs Low: $\overline{RD}$, $\overline{IORQ}$, and $\overline{CE}$. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be read. $\overline{M1}$ must be High to distinguish this cycle from an interrupt acknowledge. No additional wait states are allowed.

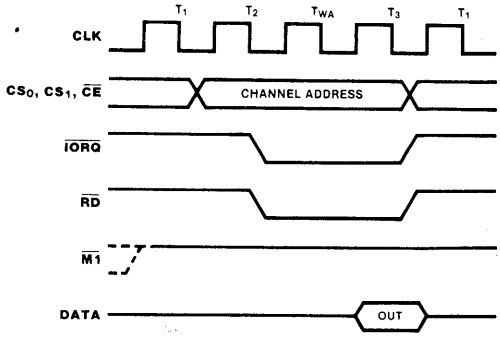


Figure 9. Read Cycle Timing

Write Cycle Timing. Figure 10 shows write cycle timing for loading control, time constant or vector words.

The CTC does not have a write signal input, so it generates one internally when the read ($\overline{RD}$) input is High during T_1. During T_2 $\overline{IORQ}$ and $\overline{CE}$ inputs are Low. $\overline{M1}$ must be High to distinguish a write cycle from an interrupt acknowledge. A 2-bit binary code at inputs CS_1 and CS_0 selects the channel to be addressed, and the word being written is placed on the Z-80 data bus. The data word is latched into the appropriate register with the rising edge of clock cycle T_3.

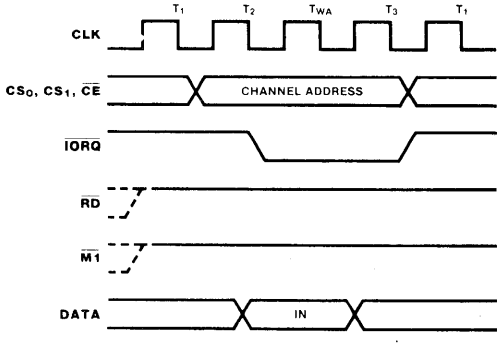


Figure 10. Write Cycle Timing

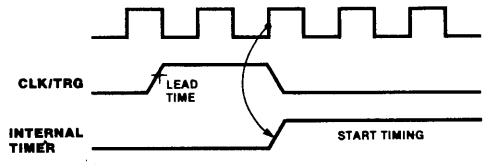


Figure 11. Timer Mode Timing

Timer Operation. In the timer mode, a CLK/TRG pulse input starts the timer (Figure 11) on the second succeeding rising edge of CLK. The trigger pulse is asynchronous, and it must have a minimum width. A minimum lead time (210 ns) is required between the active edge of the CLK/TRG and the next rising edge of CLK to enable the prescaler on the following clock edge. If the CLK/TRG edge occurs closer than this, the initiation of the timer function is delayed one clock cycle. This corresponds to the startup timing discussed in the programming section. The timer can also be started automatically if so programmed by the channel control word.

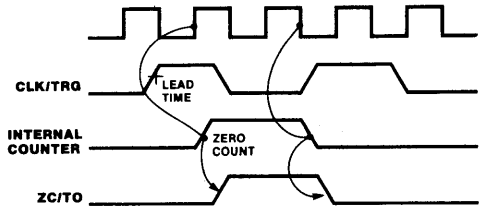


Figure 12. Counter Mode Timing

Counter Operation. In the counter mode, the CLK/TRG pulse input decrements the down-counter. The trigger is asynchronous, but the count is synchronized with CLK. For the decrement to occur on the next rising edge of CLK, the trigger edge must precede CLK by a minimum lead time as shown in Figure 12. If the lead time is less than specified, the count is delayed by one clock cycle. The trigger pulse must have a minimum width, and the trigger period must be at least twice the clock period.

The ZC/TO output occurs immediately after zero count, and follows the rising CLK edge.

Interrupt Operation

The Z-80 CTC follows the Z-80 system interrupt protocol for nested priority interrupts and return from interrupt, wherein the interrupt priority of a peripheral is determined by its location in a daisy chain. Two lines—IEI and IEO—in the CTC connect it to the system daisy chain. The device closest to the +5 V supply has the highest priority (Figure 13). For additional information on the Z-80 interrupt structure, refer to the *Z-80 CPU Product Specification* and the *Z-80 CPU Technical Manual.*

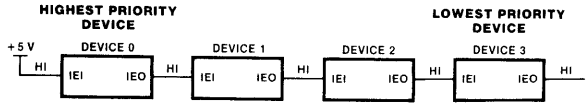


Figure 13. Daisy-Chain Interrupt Priorities

Within the Z-80 CTC, interrupt priority is predetermined by channel number: Channel 0 has the highest priority, and Channel 3 the lowest. If a device or channel is being serviced with an interrupt routine, it cannot be interrupted by a device or channel with lower priority until service is complete. Higher priority devices or channels may interrupt the servicing of lower priority devices or channels.

A Z-80 CTC channel may be programmed to request an interrupt every time its down-counter reaches zero. Note that the CPU must be programmed for interrupt mode 2. Some time after the interrupt request, the CPU sends an interrupt acknowledge. The CTC interrupt control logic determines the highest priority channel that is requesting an interrupt. Then, if the CTC IEI input is High (indicating that it has priority within the system daisy chain) it places an 8-bit interrupt vector on the system data bus. The high-order five bits of this vector were written to the CTC during the programming process; the next two bits are provided by the CTC interrupt control logic as a binary code that identifies the highest priority channel requesting an interrupt; the low-order bit is always zero.

Interrupt Acknowledge Timing. Figure 14 shows interrupt acknowledge timing. After an interrupt request, the Z-80 CPU sends an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). All channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active—about two clock cycles earlier than $\overline{IORQ}$. $\overline{RD}$ is High to distinguish this cycle from an instruction fetch.

The CTC interrupt logic determines the highest priority channel requesting an interrupt. If the CTC interrupt enable input (IEI) is High, the highest priority interrupting channel within the CTC places its interrupt vector on the data bus when $\overline{IORQ}$ goes Low. Two wait states (T_{WA}) are automatically inserted at this time to allow the daisy chain to stabilize. Additional wait states may be added.

Return from Interrupt Timing. At the end of an interrupt service routine the RETI (Return From Interrupt) instruction initializes the daisy chain enable lines for proper control of nested priority interrupt handling. The CTC decodes the 2-byte RETI code internally and determines whether it is intended for a channel being serviced. Figure 15 shows RETI timing.

If several Z-80 peripherals are in the daisy chain, IEI settles active (High) on the chip currently being serviced when the opcode ED_{16} is decoded. If the following opcode is $4D_{16}$, the peripheral being serviced is released and its IEO becomes active. Additional wait states are allowed.

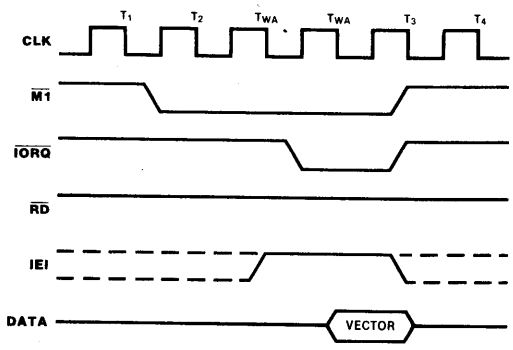


Figure 14. Interrupt Acknowledge Timing

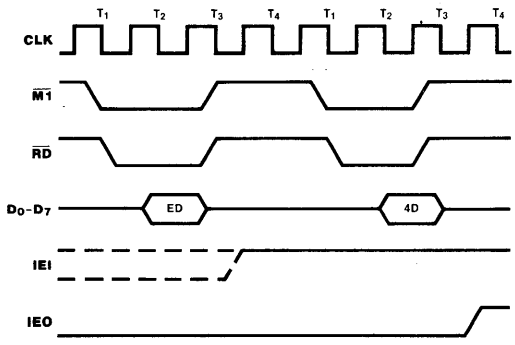


Figure 15. Return From Interrupt Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

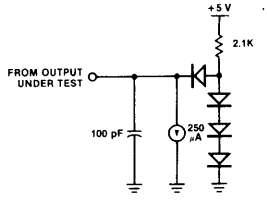
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

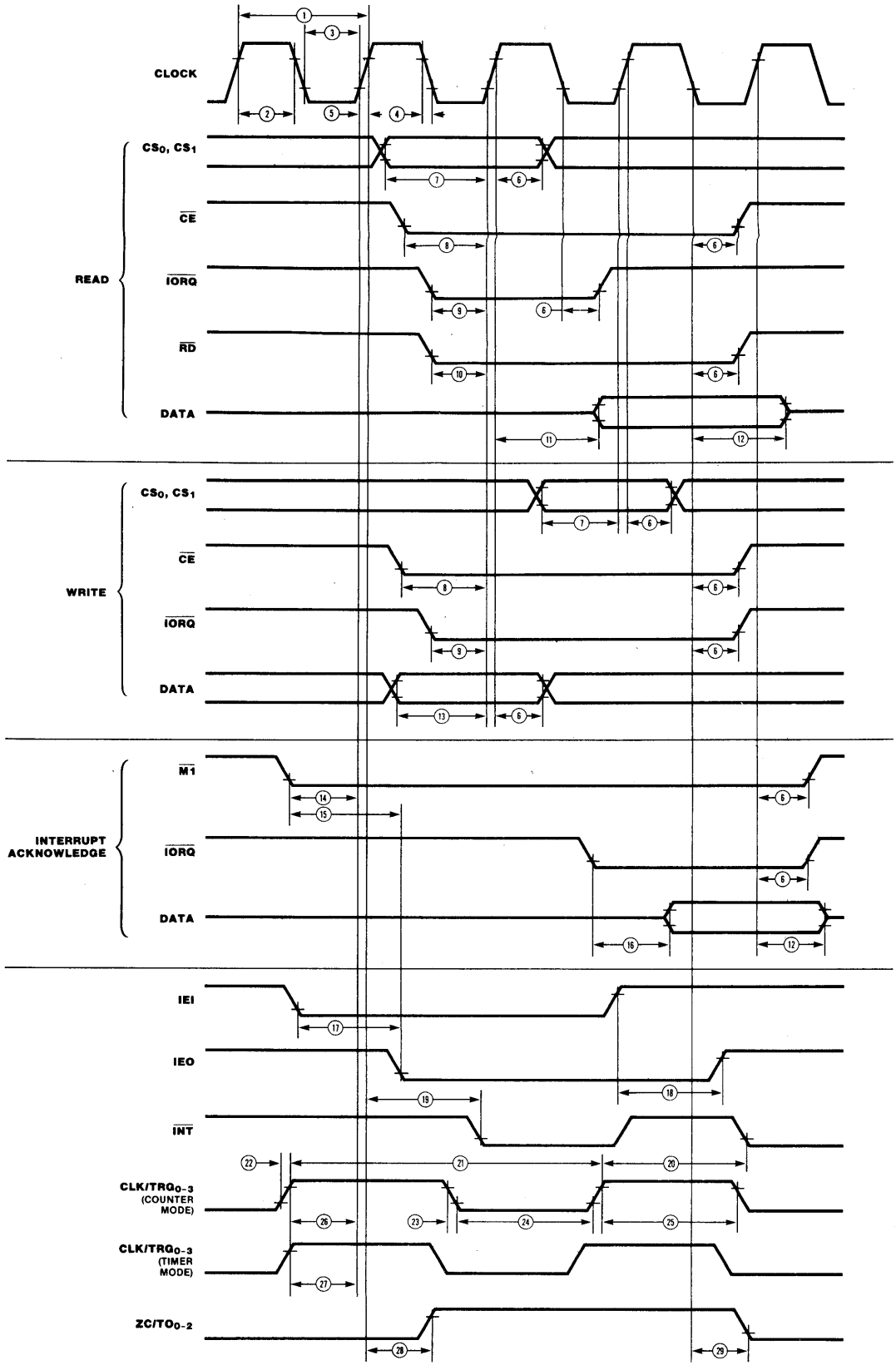


DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = −250 µA
I_{CC}	Power Supply Current		+20	mA	
I_{LI}	Input Leakage Current		±10	µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10	µA	V_{OUT} = 0.4 to V_{CC}
I_{OHD}	Darlington Drive Current	-1.5		mA	V_{OH} = 1.5 V, R_{EXT} = 390Ω

Symbol	Parameter	Max	Unit	Condition
CLK	Clock Capacitance	20	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance	5	pF	
C_{OUT}	Output Capacitance	10	pF	

T_A = 25°C, f = 1 MHz


CLOCK
READ
CS0, CS1
CE
IORQ
RD
DATA
WRITE
CS0, CS1
CE
IORQ
DATA
INTERRUPT ACKNOWLEDGE
M1
IORQ
DATA
IEI
IEO
INT
CLK/TRG0-3 (COUNTER MODE)
CLK/TRG0-3 (TIMER MODE)
ZC/TO0-2

Number	Symbol	Parameter	Z-80 CTC Min (ns)	Z-80 CTC Max (ns)	Z-80A CTC Min (ns)	Z-80A CTC Max (ns)	Z-80B CTC Min (ns)	Z-80B CTC Max (ns)	Notes*
1	TcC	Clock Cycle Time	400	[1]	250	[1]	165	[1]	
2	TwCH	Clock Width (High)	170	2000	105	2000	65	2000	
3	TwCl	Clock Width (Low)	170	2000	105	2000	65	2000	
4	TfC	Clock Fall Time		30		30		20	
5	TrC	Clock Rise Time		30		30		20	
6	Th	All Hold Times	0		0		0		
7	TsCS(C)	CS to Clock ↑ Setup Time	250		160		100		
8	TsCE(C)	$\overline{CE}$ to Clock ↑ Setup Time	200		150		100		
9	TsIO(C)	$\overline{IORQ}$ ↓ to Clock ↑ Setup Time	250		115		70		
10	TsRD(C)	$\overline{RD}$ ↓ to Clock ↑ Setup Time	240		115		70		
11	TdC(DO)	Clock ↑ to Data Out Delay		240		200		130	[2]
12	TdC(DOz)	Clock ↓ to Data Out Float Delay		230		110		90	
13	TsDI(C)	Data In to Clock ↑ Setup Time	60		50		40		
14	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		70		
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt immediately preceding $\overline{M1}$)		300		190		130	[3]
16	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		340		160		110	[2]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		190		130		100	[3]
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (After ED Decode)		220		160		110	[3]
19	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay	(TcC + 200)		(TcC + 140)		TcC + 120		[4]
20	TdCLK(INT)	CLK/TRG ↑ to $\overline{INT}$ ↓							
		tsCTR(C) satisfied	(19) + (26)		(19) + (26)		(19) + (26)		[5]
		tsCTR(C) not satisfied	(1) + (19) + (26)		(1) + (19) + (26)		(1) + (19) + (26)		[5]
21	TcCTR	CLK/TRG Cycle Time	(2TcC)		(2TcC)		2TcC		[5]
22	TrCTR	CLK/TRG Rise Time		50		50		40	
23	TfCTR	CLK/TRG Fall Time		50		50		40	
24	TwCTRl	CLK/TRG Width (Low)	200		200		120		
25	TwCTRh	CLK/TRG Width (High)	200		200		120		
26	TsCTR(Cs)	CLK/TRG ↑ to Clock ↑ Setup Time for Immediate Count	300		210		150		[5]
27	TsCTR(Ct)	CLK/TRG ↑ to Clock ↑ Setup Time for enabling of Prescaler on following clock↑	210		210		150		[4]
28	TdC(ZC/TOr)	Clock ↑ to ZC/TO ↑ Delay		260		190		140	
29	TdC(ZC/TOf)	Clock ↓ to ZC/TO ↓ Delay		190		190		140	

[A] 2.5 TcC > (n-2) TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) + TTL buffer delay, if any.
[B] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase delay by 10 ns for each 50 pF increase in loading, 200 pF maximum for data lines, and 100 pF for control lines.
[3] Increase delay by 2 ns for each 10 pF increase in loading, 100 pF maximum.
[4] Timer mode.
[5] Counter mode.
[6] RESET must be active for a minimum of 3 clock cycles.
* All timings are preliminary and subject to change.

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8430	CE	2.5 MHz	Z80 CTC (28-pin)	Z8430A	CMB	4.0 MHz	Z80A CTC (28-pin)
Z8430	CM	2.5 MHz	Same as above	Z8530A	CS	4.0 MHz	Same as above
Z8430	CMB	2.5 MHz	Same as above	Z8430A	DE	4.0 MHz	Same as above
Z8430	CS	2.5 MHz	Same as above	Z8430A	DS	4.0 MHz	Same as above
Z8430	DE	2.5 MHz	Same as above	Z8430A	PE	4.0 MHz	Same as above
Z8430	DS	2.5 MHz	Same as above	Z8430A	PS	4.0 MHz	Same as above
Z8430	PE	2.5 MHz	Same as above	Z8430B	CS	6.0 MHz	Same as above
Z8430	PS	2.5 MHz	Same as above	Z8430B	DS	6.0 MHz	Same as above
Z8430A	CE	4.0 MHz	Z80A CTC (28-pin)	Z8430B	PS	6.0 MHz	Same as above
Z8430A	CM	4.0 MHz	Same as above				

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8440
Z80® SIO Serial Input/Output Controller

Zilog

Product Specification

September 1983

Features

- Two independent full-duplex channels, with separate control and status lines for modems or other devices.
- Data rates of 0 to 500K bits/second in the x1 clock mode with a 2.5 MHz clock (Z-80 SIO), or 0 to 800K bits/second with a 4.0 MHz clock (Z-80A SIO).
- Asynchronous protocols: everything necessary for complete messages in 5, 6, 7 or 8 bits/character. Includes variable stop bits and several clock-rate multipliers; break generation and detection; parity; overrun and framing error detection.
- Synchronous protocols: everything necessary for complete bit- or byte-oriented messages in 5, 6, 7 or 8 bits/character, including IBM Bisync, SDLC, HDLC, CCITT-X.25 and others. Automatic CRC generation/checking, sync character and zero insertion/deletion, abort generation/detection and flag insertion.
- Receiver data registers quadruply buffered, transmitter registers doubly buffered.
- Highly sophisticated and flexible daisy-chain interrupt vectoring for interrupts without external logic.

General Description

The Z-80 SIO Serial Input/Output Controller is a dual-channel data communication interface with extraordinary versatility and capability. Its basic functions as a serial-to-parallel, parallel-to-serial converter/controller can be programmed by a CPU for a broad range of serial communication applications.

The device supports all common asynchronous and synchronous protocols, byte- or bit-oriented, and performs all of the functions traditionally done by UARTs, USARTs and synchronous communication controllers combined, plus additional functions traditionally performed by the CPU. Moreover, it does this on two fully-independent channels, with an exceptionally sophisticated interrupt structure that allows very fast transfers.

Full interfacing is provided for CPU or DMA

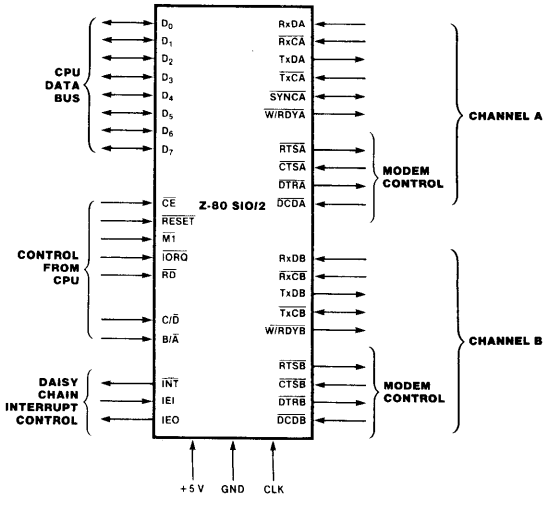


Figure 1. Z-80 SIO/2 Pin Functions

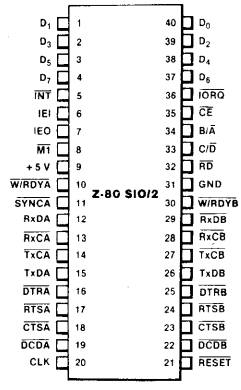


Figure 2. Z-80 SIO/2 Pin Assignments

General Description (Continued)

control. In addition to data communication, the circuit can handle virtually all types of serial I/O with fast (or slow) peripheral devices. While designed primarily as a member of the Z-80 family, its versatility makes it well suited to many other CPUs.

The Z-80 SIO is an n-channel silicon-gate depletion-load device packaged in a 40-pin plastic or ceramic DIP. It uses a single +5 V power supply and the standard Z-80 family single-phase clock.

Pin Description

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{RxC}$), Transmit Clock ($\overline{TxC}$), Data Terminal Ready ($\overline{DTR}$) and Sync ($\overline{SYNC}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- Z-80 SIO/2 lacks $\overline{SYNCB}$
- Z-80 SIO/1 lacks $\overline{DTRB}$
- Z-80 SIO/0 has all four signals, but $\overline{TxCB}$ and $\overline{RxCB}$ are bonded together

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/$\overline{D}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z-80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z-80 SIO. D_0 is the least significant bit.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffer-

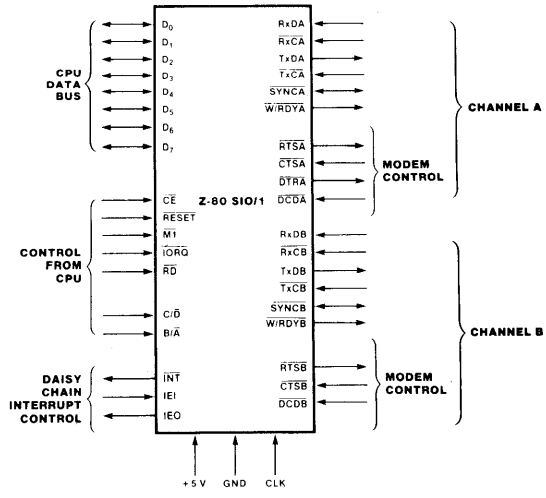


Figure 3. Z-80 SIO/1 Pin Functions

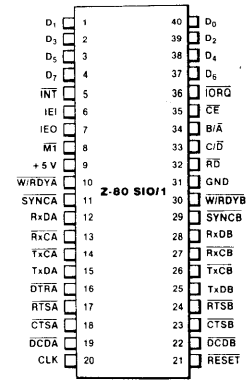


Figure 4. Z-80 SIO/1 Pin Assignments

Pin Description (Continued)

ing does not guarantee a specific noise-level margin.

$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into Z-80 SIO. They can also be programmed as general-purpose outputs.

In the Z-80 SIO/1 bonding option, $\overline{\text{DTRB}}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{\text{INT}}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{\text{INT}}$ Low.

$\overline{\text{IORQ}}$. *Input/Output Request* (input from CPU, active Low). $\overline{\text{IORQ}}$ is used in conjunction with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ to transfer commands and data between the CPU and the SIO. When $\overline{\text{CE}}$, $\overline{\text{RD}}$ and $\overline{\text{IORQ}}$ are all active, the channel selected by B/$\overline{\text{A}}$ transfers data to the CPU (a read operation). When $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ are active but $\overline{\text{RD}}$ is inactive, the channel selected by B/$\overline{\text{A}}$ is written to by the CPU with either data or control information as specified by C/$\overline{\text{D}}$. If $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{\text{M1}}$. *Machine Cycle* (input from Z-80 CPU, active Low). When $\overline{\text{M1}}$ is active and $\overline{\text{RD}}$ is also active, the Z-80 CPU is fetching an instruction from memory; when $\overline{\text{M1}}$ is active while $\overline{\text{IORQ}}$ is active, the SIO accepts $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z-80 CPU.

$\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{\text{RxC}}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the Z-80 SIO/0 bonding option, $\overline{\text{RxCB}}$ is bonded together with $\overline{\text{TxCB}}$.

$\overline{\text{RD}}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{\text{RD}}$ is active, a memory or I/O read operation is in progress. $\overline{\text{RD}}$ is used with B/$\overline{\text{A}}$, $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{\text{RESET}}$. *Reset* (input, active Low). A Low $\overline{\text{RESET}}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be

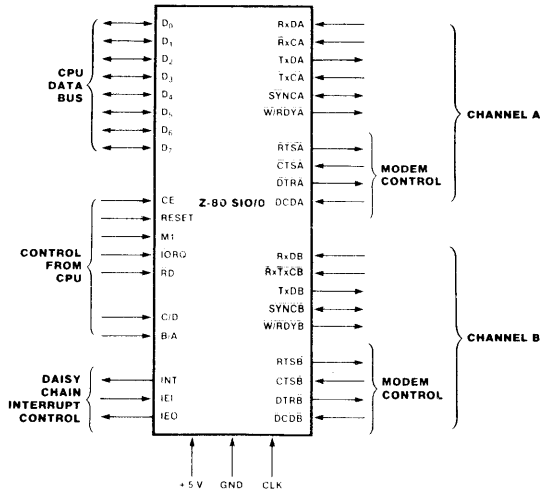


Figure 5. Z-80 SIO/0 Pin Functions

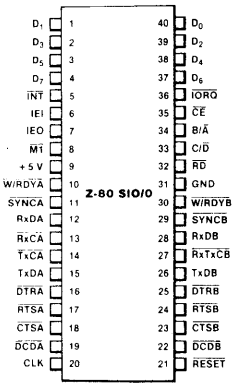


Figure 6. Z-80 SIO/0 Pin Assignments

Pin Description (Continued)

rewritten after the SIO is reset and before data is transmitted or received.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{\text{RTS}}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\text{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\text{SYNCA}}$, $\overline{\text{SYNCB}}$. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of $\overline{\text{SYNC}}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z-80 SIO/2 bonding option, $\overline{\text{SYNCB}}$ is omitted.

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z-80 SIO/0 bonding option, $\overline{\text{TxCB}}$ is bonded together with $\overline{\text{RxCB}}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{\text{TxC}}$.

$\overline{\text{W/RDYA}}$, $\overline{\text{W/RDYB}}$. *Wait/Ready A, Wait/Ready B* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

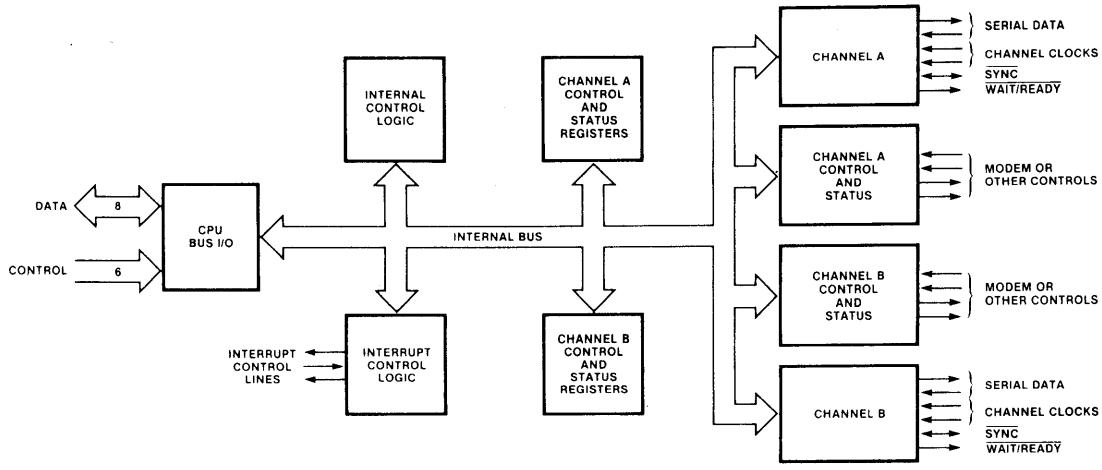


Figure 7. Block Diagram

Functional Description

The functional capabilities of the Z-80 SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data in a wide variety of data-communication protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other peripheral circuits, sharing the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

Figure 8 illustrates the conventional devices that the SIO replaces.

The first part of the following discussion covers SIO data-communication capabilities; the second part describes interactions between the CPU and the SIO.

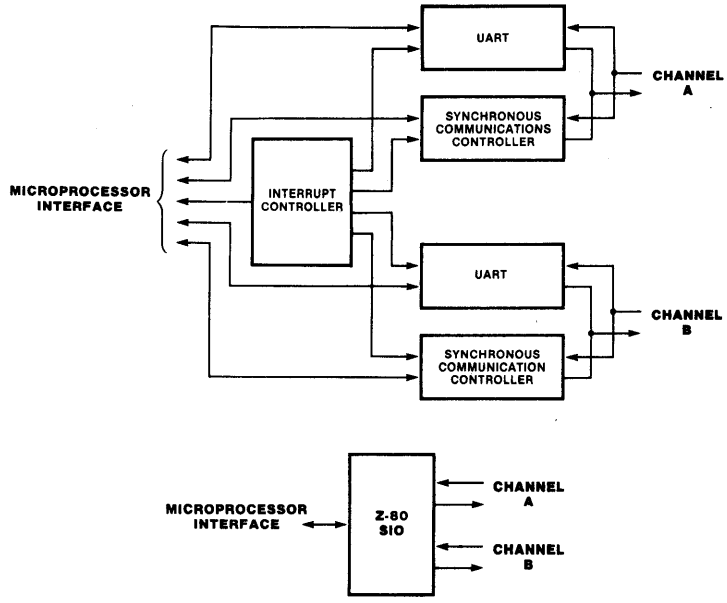


Figure 8. Conventional Devices Replaced by the Z-80 SIO

Data Communication Capabilities

The SIO provides two independent full-duplex channels that can be programmed for use in any common asynchronous or synchronous data-communication protocol. Figure 9 illustrates some of these protocols. The following is a short description of them. A more detailed explanation of these modes can be found in the *Z-80 SIO Technical Manual.*

Asynchronous Modes. Transmission and reception can be done independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 5). If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The SIO does not require symmetric transmit and receive clock signals—a feature that allows it to be used with a Z-80 CTC or many other clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

In asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The SIO supports both byte-oriented and bit-oriented synchronous communication.

Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync

Data Communication Capabilities (Continued)

characters can be removed without interrupting the CPU.

Five-, six- or seven-bit sync characters are detected with 8- or 16-bit patterns in the SIO by overlapping the larger pattern across multiple in-coming sync characters, as shown in Figure 10.

CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. The SIO can be used for interfacing to peripherals such as hard-sectored floppy disk, but it cannot generate or check CRC for IBM-compatible soft-sectored disks. The SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length.

The SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message the SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. One to eight bits per character can be sent, which allows reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame in SDLC or HDLC, and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin; an interrupt can also be programmed. The receiver can be programmed to search for frames addressed by a single byte to only a specified user-selected address or to a global broadcast address. In this mode, frames that do not match either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For transmitting data, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers.

The SIO can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SIO then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received.

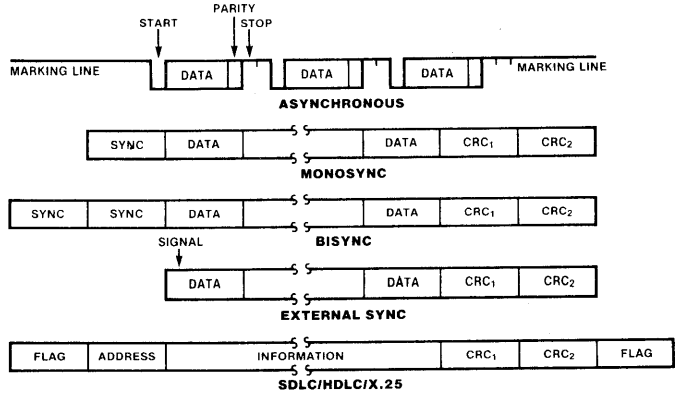


Figure 9. Some Z-80 SIO Protocols

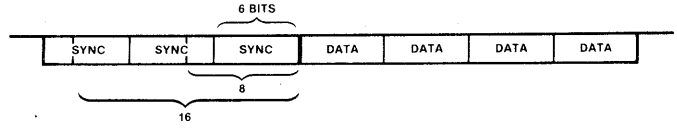


Figure 10. Six-Bit Sync Character Recognition

I/O Interface Capabilities

The SIO offers the choice of polling, interrupt (vectored or non-vectored) and block-transfer modes to transfer data, status and control information to and from the CPU. The block-transfer mode can also be implemented under DMA control.

Polling. Two status registers are updated at appropriate times for each function being performed (for example, CRC error-status valid at the end of a message). When the CPU is operated in a polling fashion, one of the SIO's two status registers is used to indicate whether the SIO has some data or needs some data. Depending on the contents of this register, the CPU will either write data, read data, or just go on. Two bits in the register indicate that a data transfer is needed. In addition, error and other conditions are indicated. The second status register (special receive conditions) does not have to be read in a polling sequence, until a character has been received. All interrupt modes are disabled when operating the device in a polled environment.

Interrupts. The SIO has an elaborate interrupt scheme to provide fast interrupt service in real-time applications. A control register and a status register in Channel B contain the interrupt vector. When programmed to do so, the SIO can modify three bits of the interrupt vector in the status register so that it points directly to one of eight interrupt service routines in memory, thereby servicing conditions in both channels and eliminating most of the needs for a status-analysis routine.

Transmit interrupts, receive interrupts and external/status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control, with Channel A having a higher priority than Channel B, and with receive, transmit and external/status interrupts prioritized in that order within each channel. When the transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) The receiver can interrupt the CPU in one of two ways:

- Interrupt on first received character
- Interrupt on all received characters

Interrupt-on-first-received-character is typically used with the block-transfer mode. Interrupt-on-all-received-characters has the option of modifying the interrupt vector in the event of a parity error. Both of these interrupt modes will also interrupt under special receive conditions on a character or message basis (end-of-frame interrupt in SDLC, for example). This means that the special-receive condition can cause an interrupt only if the interrupt-on-first-received-character or interrupt-on-all-received-characters mode is selected. In interrupt-on-first-received-character, an interrupt can occur from special-receive conditions (except parity error) after the first-received-character interrupt (example: receive-overrun interrupt).

The main function of the external/status interrupt is to monitor the signal transitions of the Clear To Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$) and Synchronization ($\overline{SYNC}$) pins (Figures 1 through 6). In addition, an external/status interrupt is also caused by a CRC-sending condition or by the detection of a break sequence (asynchronous mode) or abort sequence (SDLC mode) in the data stream. The interrupt caused by the break/abort sequence allows the SIO to interrupt when the break/abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the break/abort condition in external logic.

I/O Interface Capabilities (Continued)

In a Z-80 CPU environment (Figure 11), SIO interrupt vectoring is "automatic": the SIO passes its internally-modifiable 8-bit interrupt vector to the CPU, which adds an additional 8 bits from its interrupt-vector (I) register to form the memory address of the interrupt-routine table. This table contains the address of the beginning of the interrupt routine itself. The process entails an indirect transfer of CPU control to the interrupt routine, so that the next instruction executed after an interrupt acknowledge by the CPU is the first instruction of the interrupt routine itself.

CPU/DMA Block Transfer. The SIO's block-transfer mode accommodates both CPU block transfers and DMA controllers (Z-80 DMA or other designs). The block-transfer mode uses the Wait/Ready output signal, which is selected with three bits in an internal control register. The Wait/Ready output signal can be programmed as a $\overline{WAIT}$ line in the CPU block-transfer mode or as a $\overline{READY}$ line in the DMA block-transfer mode.

To a DMA controller, the SIO $\overline{READY}$ output indicates that the SIO is ready to transfer data to or from memory. To the CPU, the $\overline{WAIT}$ output indicates that the SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

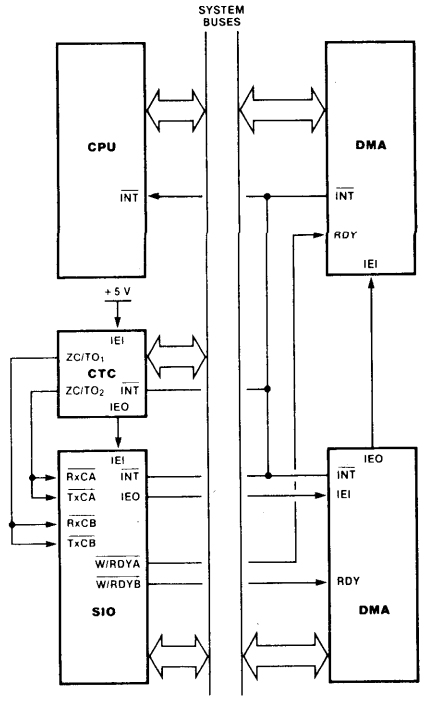


Figure 11. Typical Z-80 Environment

Internal Structure

The internal structure of the device includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains its own set of control and status (write and read) registers, and control and status logic that provides the interface to modems or other external devices.

The registers for each channel are designated as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through another 8-bit register (Read Register 2) in Channel B. The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

Read Register Functions	
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Write Register Functions	
WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

Internal Structure (Continued)

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs, Clear To Send ($\overline{CTS}$) and Data Carrier Detect ($\overline{DCD}$), are monitored by the external control and status logic under program control. All external control-and-status-logic signals are general-purpose in nature and can be used for functions other than modem control.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 12 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in asynchronous modes—the character length.

The transmitter has an 8-bit transmit data buffer register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync-character buffers or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD).

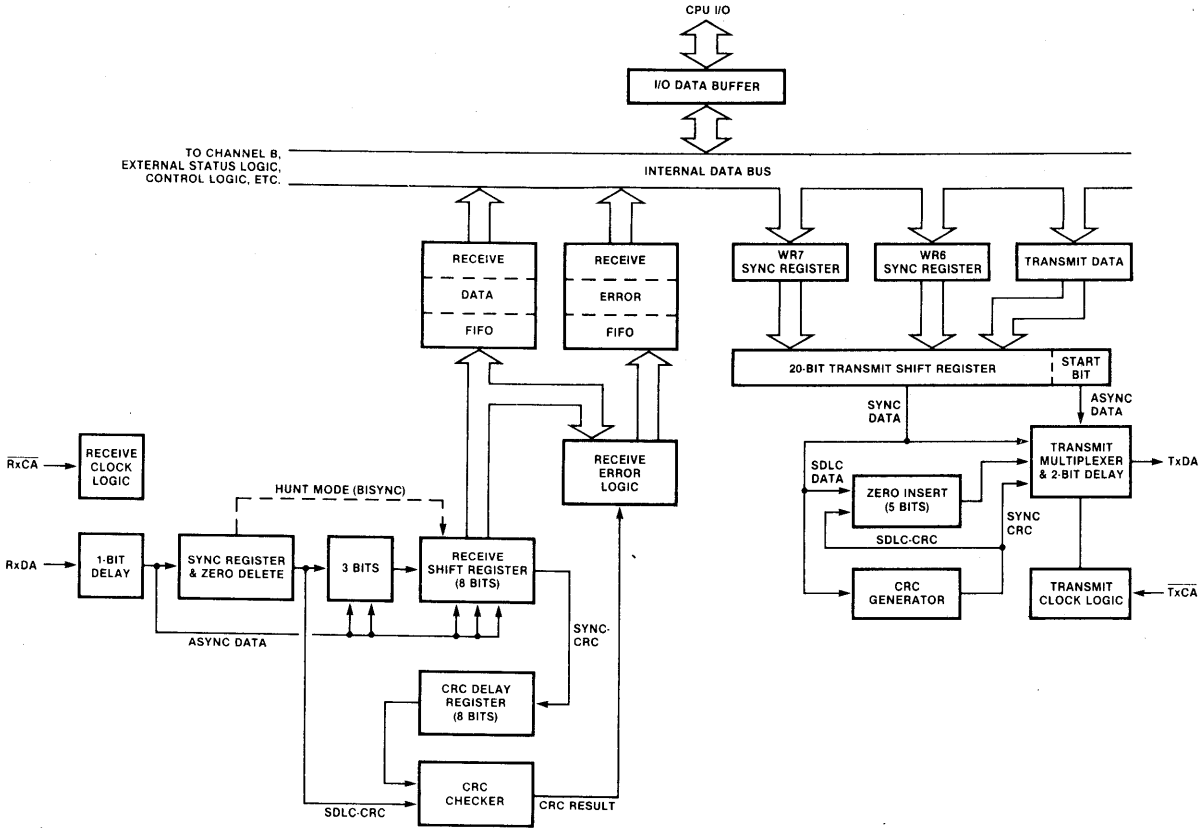


Figure 12. Transmit and Receive Data Path (Channel A)

Programming

The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\overline{A}$) and the control/data input (C/$\overline{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

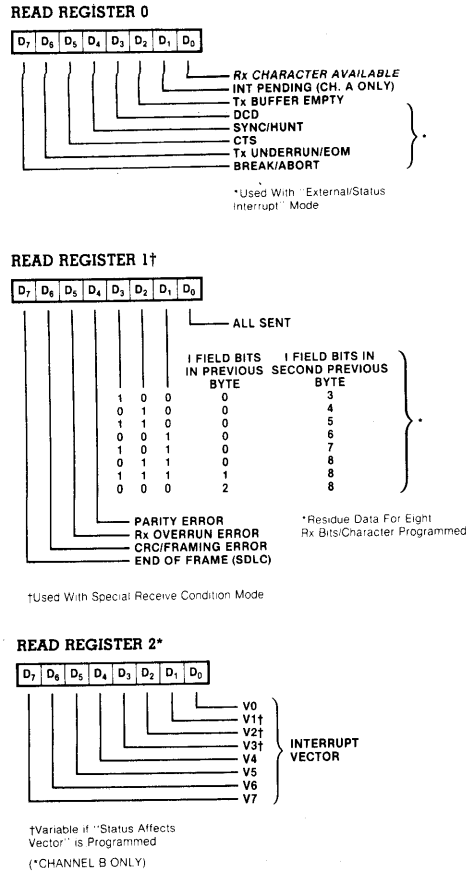


Figure 13. Read Register Bit Functions

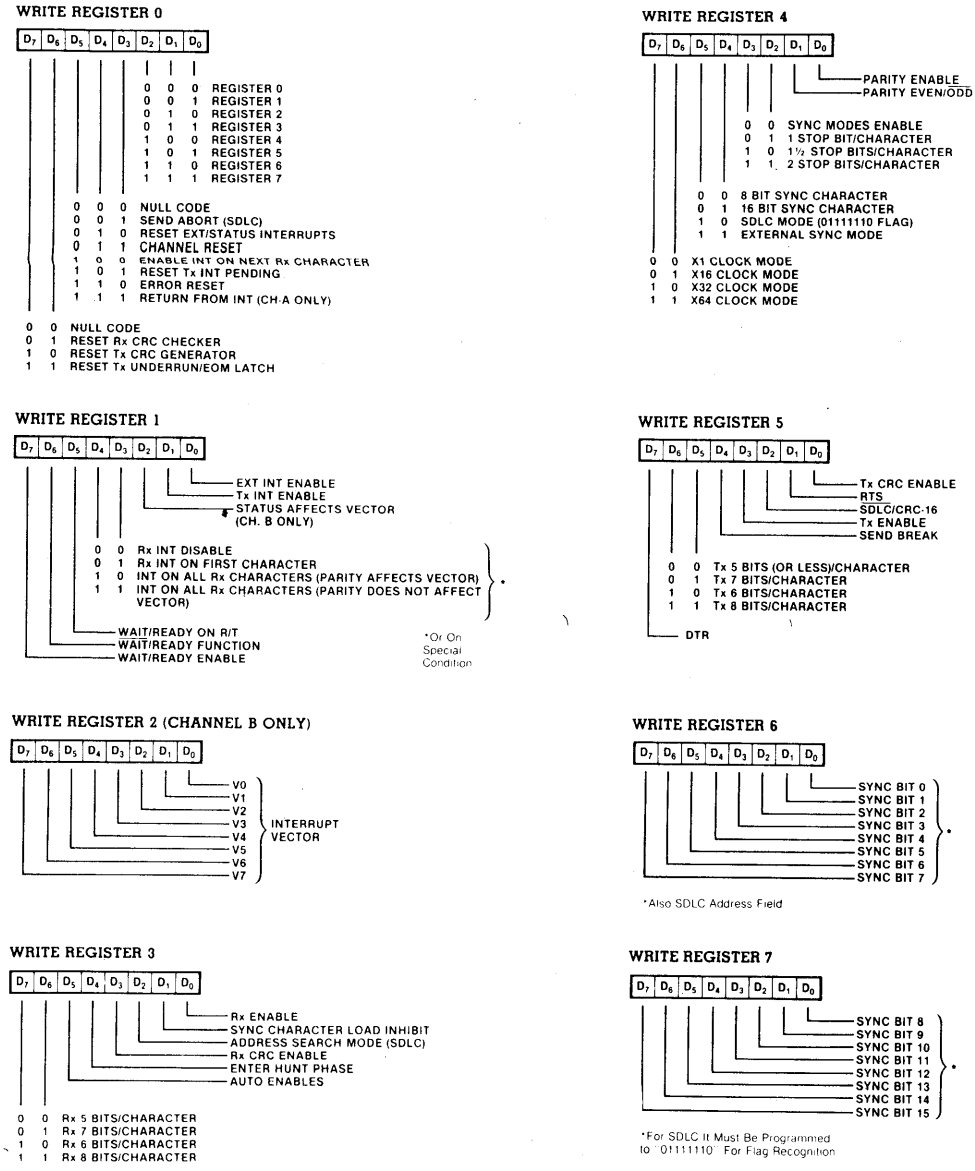


Figure 14. Write Register Bit Functions

Timing

The SIO must have the same clock as the CPU (same phase and frequency relationship, not necessarily the same driver).

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a data or status byte from the SIO are illustrated in Figure 15.

Write Cycle. Figure 16 illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a data or control byte into the SIO.

Interrupt-Acknowledge Cycle. After receiving an interrupt-request signal from an SIO ($\overline{INT}$ pulled Low), the Z-80 CPU sends an interrupt-acknowledge sequence ($\overline{M1}$ Low, and $\overline{IORQ}$ Low a few cycles later) as in Figure 17.

The SIO contains an internal daisy-chained interrupt structure for prioritizing nested interrupts for the various functions of its two channels, and this structure can be used within an external user-defined daisy chain that prioritizes several peripheral circuits.

The IEI of the highest-priority device is terminated High. A device that has an interrupt pending or under service forces its IEO Low. For devices with no interrupt pending or under service, IEO = IEI.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Return From Interrupt Cycle. Figure 18 illustrates the return from interrupt cycle. Normally, the Z-80 CPU issues a RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch in the SIO to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy-chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt-acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to the *Z-80 CPU Product Specification.*

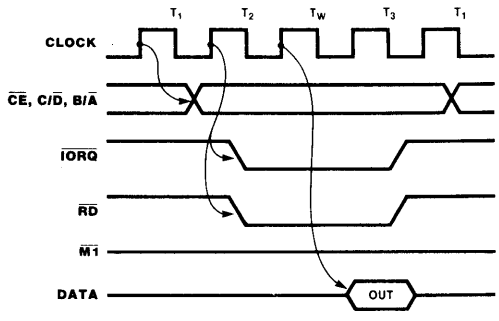


Figure 15. Read Cycle

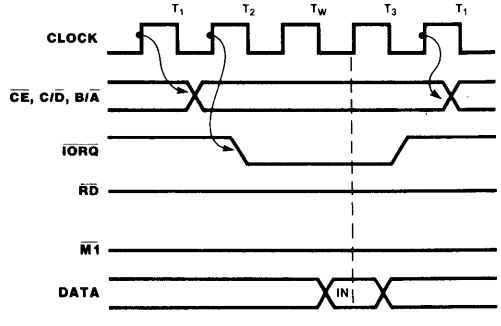


Figure 16. Write Cycle

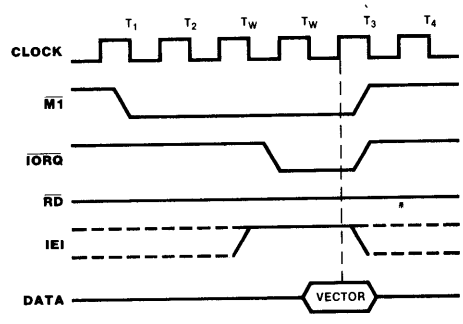


Figure 17. Interrupt Acknowledge Cycle

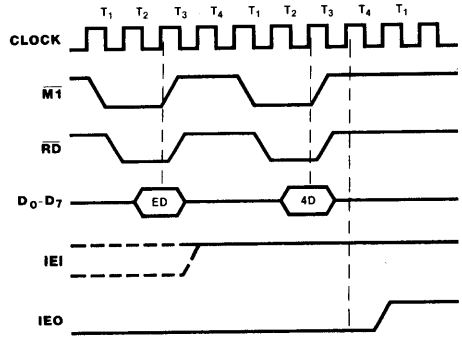


Figure 18. Return from Interrupt Cycle

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

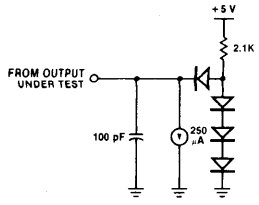
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.



The product number for each operating temperature range can be found in the ordering information included in the product specification (see 1982/83 Zilog Data Book, document number 00-2034-02).

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} −0.6	V_{CC} +0.3	V	
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = −250 µA
I_{LI}	Input Leakage Current		±10	µA	V_{IN} = 0 to V_{CC}
I_{OL}	3-State Output Leakage Current in Float		±10	µA	V_{OUT} = 0.4 V to V_{CC}
$I_{L(SY)}$	$\overline{SYNC}$ Pin Leakage Current		+10/−40	µA	$0 < V_{IN} < V_{CC}$
I_{CC}	Power Supply Current		30	mA	

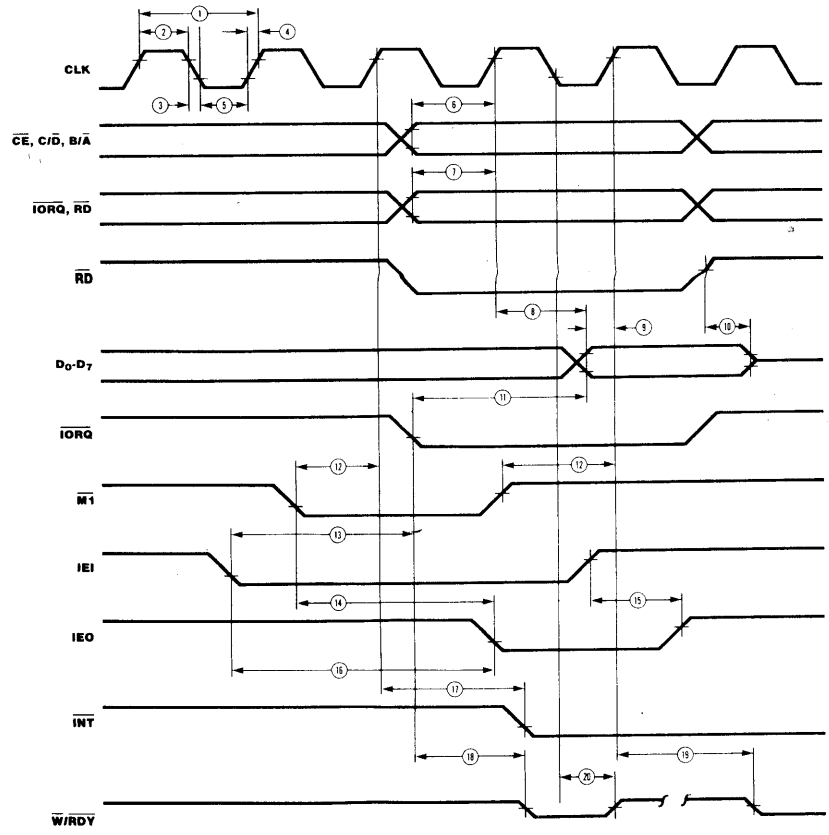
Over specified temperature and voltage range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured
C_{IN}	Input Capacitance		5	pF	pins returned
C_{OUT}	Output Capacitance		10	pF	to ground

Over specified temperature range; f = 1MHz

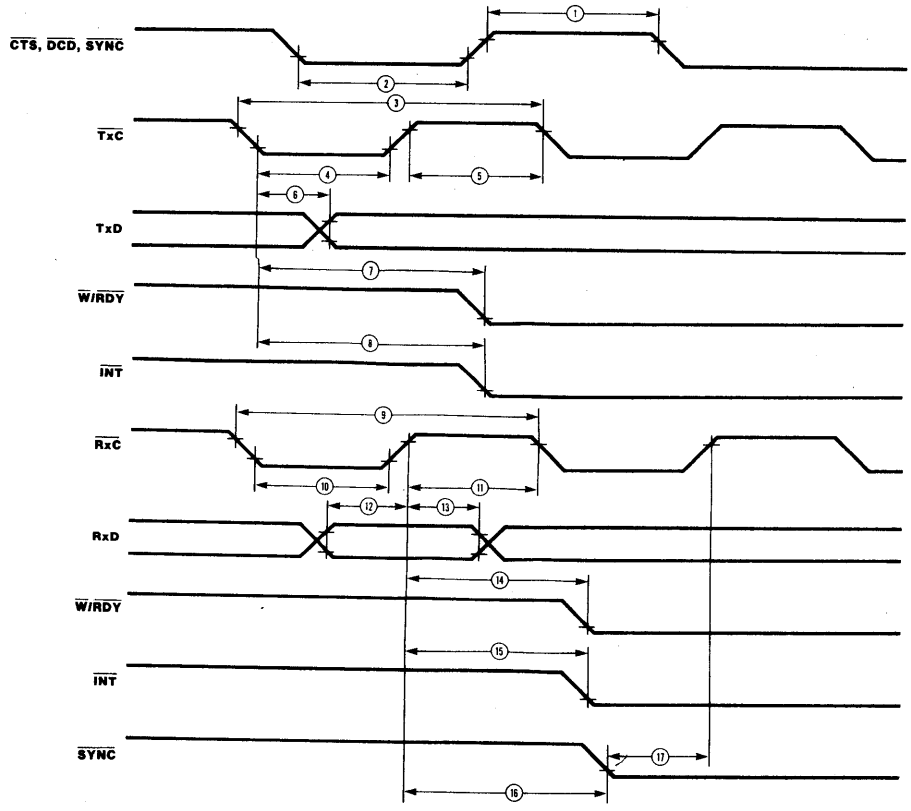
AC Electrical Characteristics



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO*† Min	Z-80B SIO*† Max
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000
3	TfC	Clock Fall Time		30		30		15
4	TrC	Clock Rise Time		30		30		15
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000
6	TsAD(C)	CE, C/D, B/A to Clock ↑ Setup Time	160		145		60	
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60	
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	50		50		30	
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		100
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75	
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	200		140		120	
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		300		190		160
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		200		200		150
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W}/\overline{RDY}$ ↓ Delay Wait Mode)		300		210		175
19	TdC(W/RR)	Clock ↑ to $\overline{W}/\overline{RDY}$ ↓ Delay (Ready Mode)		120		120		100
20	TdC(W/RWz)	Clock ↓ to $\overline{W}/\overline{RDY}$ Float Delay (Wait Mode)		150		130		110
21	Th	Any unspecified Hold when Setup is specified	0		0		0	

* Z-80 SIO timings are preliminary and subject to change.
† Units in nanoseconds (ns).

AC Electrical Characteristics (Continued)



Number	Symbol	Parameter	Z-80 SIO Min	Z-80 SIO Max	Z-80A SIO Min	Z-80A SIO Max	Z-80B SIO[1] Min	Z-80B SIO[1] Max	Notes†
1	TwPh	Pulse Width (High)	200		200		200		2
2	TwPl	Pulse Width (Low)	200		200		200		2
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	400	∞	330	∞	2
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	180	∞	100	∞	2
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	180	∞	100	∞	2
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay (x1 Mode)		400		300		220	2
7	TdTxC(W/RRf)	$\overline{TxC}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	5	9	5	9	5	9	3
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	5	9	3
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	400	∞	330	∞	2
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	180	∞	100	∞	2
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	180	∞	100	∞	2
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		0		2
13	ThRxD(RxC)	RxC ↑ to RxD Hold Time (x1 Mode)	140		140		100		2
14	TdRxC(W/RRf)	$\overline{RxC}$ ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	10	13	10	13	10	13	3
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↓ Delay	10	13	10	13	10	13	3
16	TdRxC(SYNC)	RxC ↑ to $\overline{SYNC}$ ↓ Delay (Output Modes)	4	7	4	7	4	7	3
17	TsSYNC(RxC)	$\overline{SYNC}$ ↓ to $\overline{RxC}$ ↑ Setup (External Sync Modes)	-100		-100		-100		2

NOTES:
† In all modes, the System Clock rate must be at least five times the maximum data rate.
1. Z-80B SIO timings are preliminary and subject to change.
2. Units in nanoseconds (ns).
3. Units equal to System Clock Periods.

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8440	CE,CM	2.5 MHz	Z80 SIO/0 (40-pin)
Z8440	CMB,CS	2.5 MHz	Same as above
Z8440	DE,DS	2.5 MHz	Same as above
Z8440	PE,PS	2.5 MHz	Same as above
Z8440A	CE,CM	4.0 MHz	Z80A SIO/0 (40-pin)
Z8440A	CMB,CS	4.0 MHz	Same as above
Z8440A	DE,DS	4.0 MHz	Same as above
Z8440A	PE,PS	4.0 MHz	Same as above
Z8440B	CS	6.0 MHz	Z80B SIO/0 (40-pin)
Z8440B	DS	6.0 MHz	Same as above
Z8440B	PS	6.0 MHz	Same as above
Z8441	CE,CM	2.5 MHz	Z80 SIO/1 (40-pin)
Z8441	CMB,CS	2.5 MHz	Same as above
Z8441	DE,DS	2.5 MHz	Same as above
Z8441	PE,PS	2.5 MHz	Same as above
Z8441A	CE,CM	4.0 MHz	Z80A SIO/1 (40-pin)
Z8441A	CMB,CS	4.0 MHz	Same as above
Z8441A	DE,DS	4.0 MHz	Z80B SIO/1 (40-pin)
Z8441A	PE,PS	4.0 MHz	Same as above
Z8441B	CS	6.0 MHz	Z80B SIO/1 (40-pin)
Z8441B	DS	6.0 MHz	Same as above
Z8441B	PS	6.0 MHz	Same as above
Z8442	CE,CM	2.5 MHz	Z80 SIO/2 (40-pin)
Z8442	CMB,CS	2.5 MHz	Same as above
Z8442	DE,DS	2.5 MHz	Same as above
Z8442	PE,PS	2.5 MHz	Same as above
Z8442A	CE,CM	4.0 MHz	Z80A SIO/2 (40-pin)
Z8442A	CMB,CS	4.0 MHz	Same as above
Z8442A	DE,DS	4.0 MHz	Same as above
Z8442A	PE,PS	4.0 MHz	Same as above
Z8442B	CS	6.0 MHz	Z80B SIO/2 (40-pin)
Z8442B	DS	6.0 MHz	Same as above
Z8442B	PS	6.0 MHz	Same as above

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8470 Z80® DART Dual Asynchronous Receiver/Transmitter

Zilog

Product Specification

September 1983

Features

- Two independent full-duplex channels with separate modem controls. Modem status can be monitored.
- In x1 clock mode, data rates are 0 to 500K bits/second with a 2.5 MHz clock, or 0 to 800K bits/second with a 4.0 MHz clock.
- Receiver data registers are quadruply buffered; the transmitter is doubly buffered.
- Programmable options include 1, 1½ or 2 stop bits; even, odd or no parity; and x1, x16, x32 and x64 clock modes.
- Break generation and detection as well as parity-, overrun- and framing-error detection are available.
- Interrupt features include a programmable interrupt vector, a "status affects vector" mode for fast interrupt processing, and the standard Z-80 peripheral daisy-chain interrupt structure that provides automatic interrupt vectoring with no external logic.
- On-chip logic for ring indication and carrier-detect status.

Description

The Z-80 DART (Dual-Channel Asynchronous Receiver/Transmitter) is a dual-channel multi-function peripheral component that satisfies a wide variety of asynchronous serial data communications requirements in microcomputer systems. The Z-80 DART is used as a serial-to-parallel, parallel-to-serial converter/controller in asynchronous applications. In addition, the device also provides modem controls for both channels. In applications where modem controls are not needed, these lines can be used for general-purpose I/O.

Zilog also offers the Z-80 SIO, a more versatile device that provides synchronous (Bisync, HDLC and SDLC) as well as asynchronous operation.

The Z-80 DART is fabricated with n-channel silicon-gate depletion-load technology, and is packaged in a 40-pin plastic or ceramic DIP.

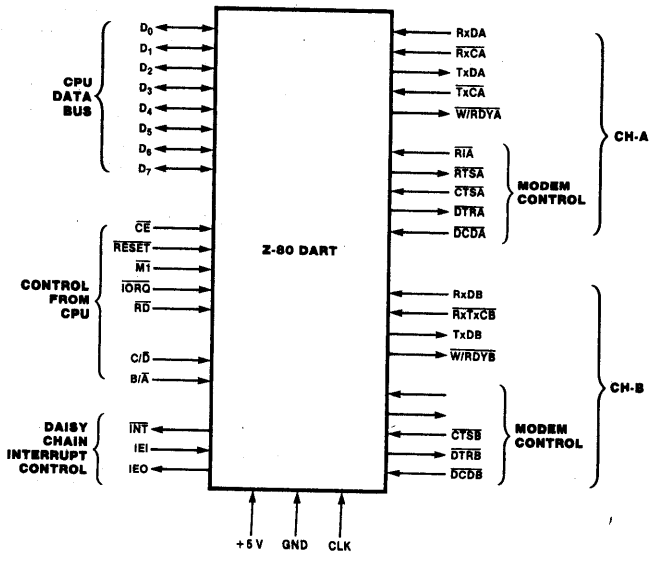


Figure 1. Z80 DART Pin Functions

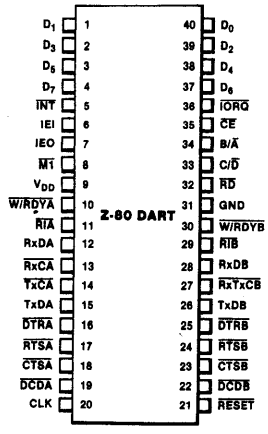


Figure 2. Pin Assignments

Pin Description

B/$\overline{A}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the Z-80 DART.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input specifies the type of information (control or data) transferred on the data bus between the CPU and the Z-80 DART.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low at this input enables the Z-80 DART to accept command or data input from the CPU during a write cycle, or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The Z-80 DART uses the standard Z-80 single-phase system clock to synchronize internal signals.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals.

D_0-D_7. *System Data Bus* (bidirectional, 3-state) transfers data and commands between the CPU and the Z-80 DART.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the Z-80 DART is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered.

$\overline{DTRA}$, $\overline{DTRB}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be programmed as general-purpose outputs.

IEI. *Interrupt Enable In* (input, active High) is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this Z-80 DART. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the Z-80 DART is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{M1}$. *Machine Cycle One* (input from Z-80 CPU, active Low). When $\overline{M1}$ and $\overline{RD}$ are both active, the Z-80 CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the Z-80 DART accepts $\overline{M1}$ and $\overline{IORQ}$ as an interrupt acknowledge if the Z-80 DART is the highest priority device that has interrupted the Z-80 CPU.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the Z-80 DART. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$.

$\overline{RxCA}$, $\overline{RxCB}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{RxC}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate.

$\overline{RD}$. *Read Cycle Status.* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress.

RxDA, RxDB. *Receive Data* (inputs, active High).

$\overline{RESET}$. *Reset* (input, active Low). Disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts.

$\overline{RIA}$, $\overline{RIB}$. *Ring Indicator* (inputs, Active Low). These inputs are similar to $\overline{CTS}$ and $\overline{DCD}$. The Z-80 DART detects both logic level transitions and interrupts the CPU. When not used in switched-line applications, these inputs can be used as general-purpose inputs.

$\overline{RTSA}$, $\overline{RTSB}$. *Request to Send* (outputs, active Low). When the RTS bit is set, the $\overline{RTS}$ output goes Low. When the RTS bit is reset, the output goes High after the transmitter empties.

$\overline{TxCA}$, $\overline{TxCB}$. *Transmitter Clocks* (inputs). TxD changes on the falling edge of $\overline{TxC}$. The Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered. Both the Receiver and Transmitter Clocks may be driven by the Z-80 CTC Counter Time Circuit for programmable baud rate generation.

TxDA, TxDB. *Transmit Data* (outputs, active High).

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *Wait/Ready* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the Z-80 DART data rate. The reset state is open drain.

Functional Description

The functional capabilities of the Z-80 DART can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of asynchronous data communications protocols; as a Z-80 family peripheral, it interacts with the Z-80 CPU and other Z-80 peripheral circuits, and shares the data, address and control buses, as well as being a part of the Z-80 interrupt structure. As a peripheral to other microprocessors, the Z-80 DART offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

The first part of the following functional description introduces Z-80 DART data communications capabilities; the second part describes the interaction between the CPU and the Z-80 DART.

The Z-80 DART offers RS-232 serial communications support by providing device signals for external modem control. In addition to dual-channel Request To Send, Clear To Send, and Data Carrier Detect ports, the Z-80 DART also features a dual channel Ring Indicator (RIA, RIB) input to facilitate local/remote or station-to-station communication capability.

Communications Capabilities. The Z-80 DART provides two independent full-duplex channels for use as an asynchronous receiver/transmitter. The following is a short description of receiver/transmitter capabilities. For more details, refer to the Asynchronous Mode section of the *Z-80 SIO Technical Manual.* The Z-80 DART offers transmission and reception of five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one and a half or two stop bits per character and can provide a break output at any time. The receiver break detection logic interrupts the CPU both at the start and end of a received break. Reception is protected from spikes by a transient spike rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the Receive Data input. If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the character on which they occurred. Vectored interrupts allow fast servicing of interrupting conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The Z-80 DART does not require symmetric Transmit and Receive Clock signals—a feature that allows it to be used with a Z-80 CTC or any other clock source. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the Receive and Transmit Clock inputs. When using Channel B, the bit rates for transmit and receive operations must be the same because $\overline{RxC}$ and $\overline{TxC}$ are bonded together ($\overline{RxTxCB}$).

I/O Interface Capabilities. The Z-80 DART offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

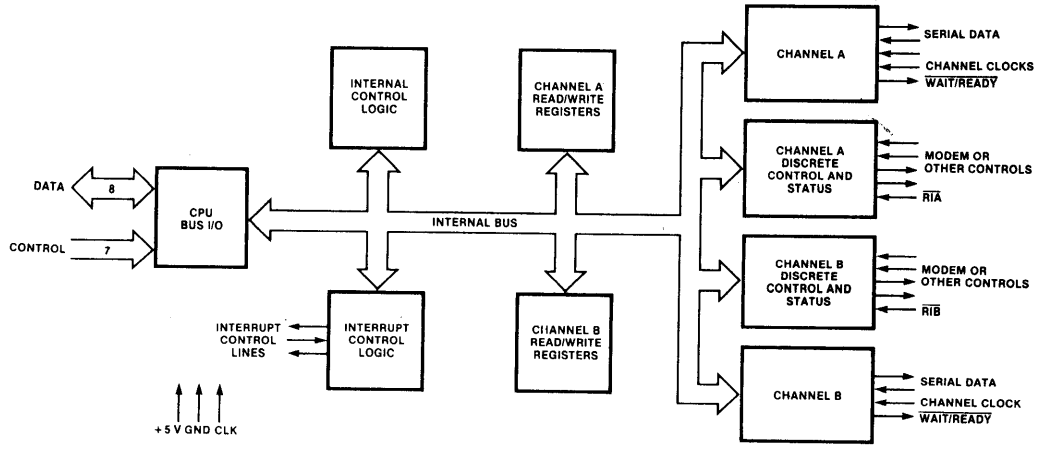


Figure 3. Block Diagram

Functional Description (Continued)

POLLING. There are no interrupts in the Polled mode. Status registers RR0 and RR1 are updated at appropriate times for each function being performed. All the interrupt modes of the Z-80 DART must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a data transfer is needed. The status also indicates Error or other special status conditions (see "Z-80 DART Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in RR1 are accompanied by a Receive Character Available status in RR0.

INTERRUPTS. The Z-80 DART offers an elaborate interrupt scheme that provides fast interrupt response in real-time applications. As a member of the Z-80 family, the Z-80 DART can be daisy-chained along with other Z-80 peripherals for peripheral interrupt-priority resolution. In addition, the internal interrupts of the Z-80 DART are nested to prioritize the various interrupts generated by Channels A and B. Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To eliminate the necessity of writing a status analysis routine, the Z-80 DART can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in RR2 is modified according to the assigned priority of the various interrupting conditions.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer *becoming* empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on the first received character
- Interrupt on all received characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters can optionally modify the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character basis. The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{RI}$ pins; however, an External/Status interrupt is also caused by the detection of a Break sequence in the data stream. The interrupt caused by the Break sequence has a special feature that allows the Z-80 DART to interrupt when the Break sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Break condition.

CPU/DMA BLOCK TRANSFER. The Z-80 DART provides a Block Transfer mode to accommodate CPU block transfer functions and DMA block transfers (Z-80 DMA or other designs). The Block Transfer mode uses the $\overline{W/RDY}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{W/RDY}$ output can be defined under software control as a Wait line in the CPU Block Transfer mode or as a Ready line in the DMA Block Transfer mode.

To a DMA controller, the Z-80 DART Ready output indicates that the Z-80 DART is ready to transfer data to or from memory. To the CPU, the Wait output indicates that the Z-80 DART is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

Internal Architecture

The device internal structure includes a Z-80 CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains read and write registers, and discrete control and status logic that provides the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated as follows:

WR0-WR5 — Write Registers 0 through 5
RR0-RR2 — Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process.

The logic for both channels provides formats, bit synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send ($\overline{CTS}$), Data Carrier Detect ($\overline{DCD}$) and Ring Indicator ($\overline{RI}$) are monitored by the control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/Status interrupts are prioritized in that order within each channel.

Data Path. The transmit and receive data path illustrated for Channel A in Figure 4 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service a Receive Character Available interrupt in a high-speed data transfer.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus, and a 9-bit transmit shift register that is loaded from the transmit data register.

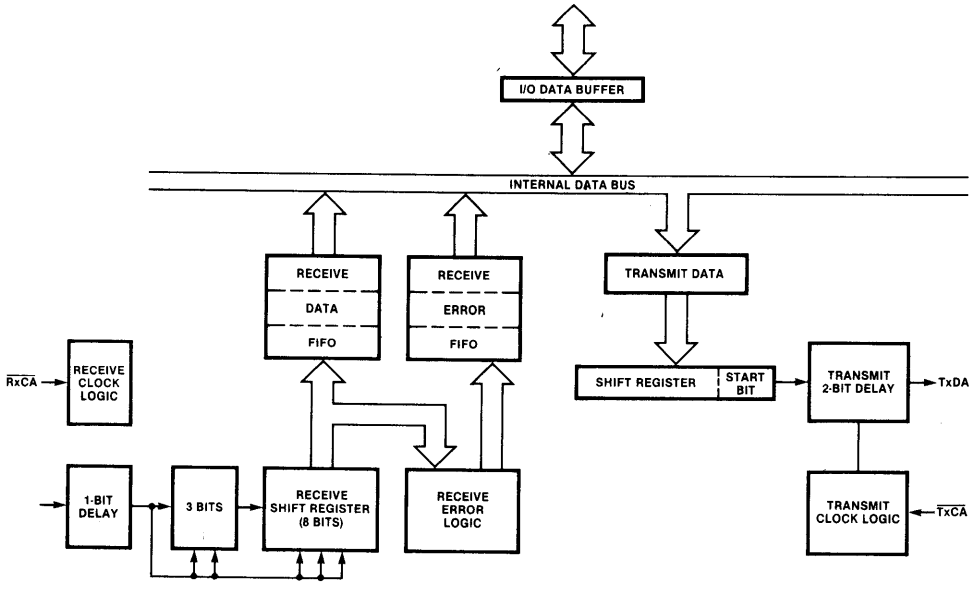


Figure 4. Data Path

Read, Write and Interrupt Timing

Read Cycle. The timing signals generated by a Z-80 CPU input instruction to read a Data or Status byte from the Z-80 DART are illustrated in Figure 5a.

Write Cycle. Figure 5b illustrates the timing and data signals generated by a Z-80 CPU output instruction to write a Data or Control byte into the Z-80 DART.

Interrupt Acknowledge Cycle. After receiving an Interrupt Request signal ($\overline{INT}$ pulled Low), the Z-80 CPU sends an Interrupt Acknowledge signal ($\overline{M1}$ and $\overline{IORQ}$ both Low). The daisy-chained interrupt circuits determine the highest priority interrupt requestor. The IEI of the highest priority peripheral is terminated High. For any peripheral that has no interrupt pending or under service, IEO = IEI. Any peripheral that does have an interrupt pending or under service forces its IEO Low.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented *from changing while* $\overline{M1}$ *is Low. When* $\overline{IORQ}$ *is* Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

Refer to the *Z-80 SIO Technical Manual* for additional details on the interrupt daisy chain and interrupt nesting.

Return From Interrupt Cycle. Normally, the Z-80 CPU issues an RETI (Return From Interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch to terminate the interrupt that has just been processed.

When used with other CPUs, the Z-80 DART allows the user to return from the interrupt cycle with a special command called "Return From Interrupt" in Write Register 0 of Channel A. This command is interpreted by the Z-80 DART in exactly the same way it would interpret an RETI command on the data bus.

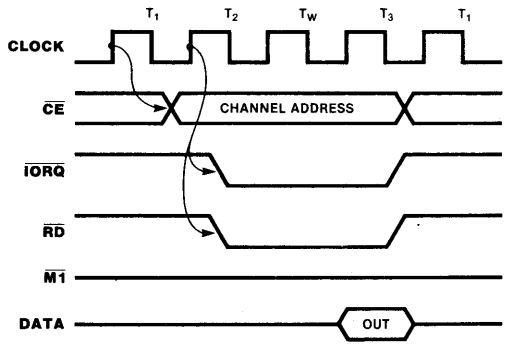


Figure 5a. Read Cycle

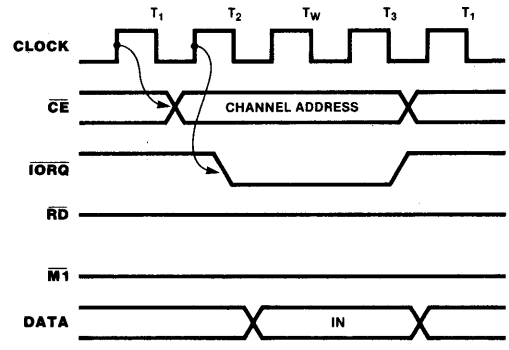


Figure 5b. Write Cycle

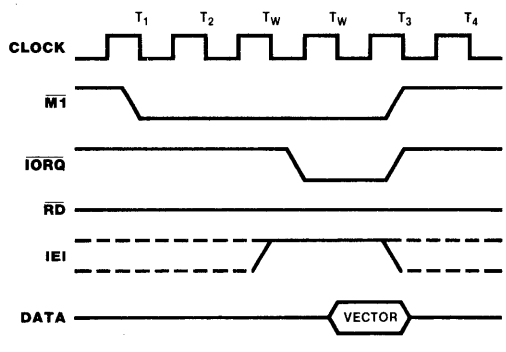


Figure 5c. Interrupt Acknowledge Cycle

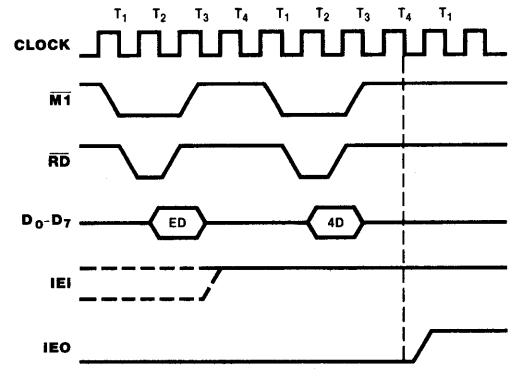


Figure 5d. Return from Interrupt Cycle

Z-80 DART Programming

To program the Z-80 DART, the system program first issues a series of commands that initialize the basic mode and then other commands that qualify conditions within the selected mode. For example, the character length, clock rate, number of stop bits, even or odd parity are first set, then the Interrupt mode and, finally, receiver or transmitter enable.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/$\overline{A}$) and the Control/Data input (C/$\overline{D}$) are the command structure addressing controls, and are normally controlled by the CPU address bus.

Write Registers. The Z-80 DART contains six registers (WR0-WR5) in each channel that are programmed separately by the system program to configure the functional personality of the channels (Figure 4). With the exception of WR0, programming the write registers requires two bytes. The first byte contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the Z-80 DART.

WR0 is a special case in that all the basic commands (CMD_0-CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This means that a register cannot be pointed to in the same operation as a channel reset.

Write Register Functions

WR0	Register pointers, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls

Read Registers. The Z-80 DART contains three registers (RR0-RR2) that can be read to obtain the status information for each channel (except for RR2, which applies to Channel B only). The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing an input instruction, the contents of the addressed read register can be read by the CPU.

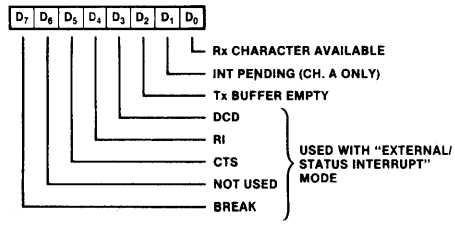
The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Read Register Functions

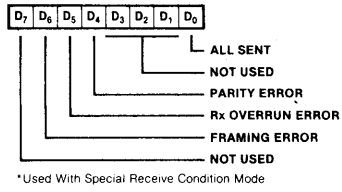
RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Z-80 DART Read and Write Registers

READ REGISTER 0

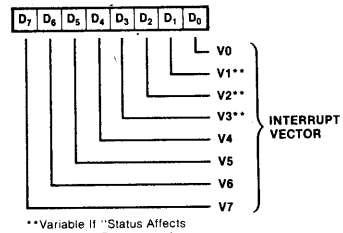


READ REGISTER 1*



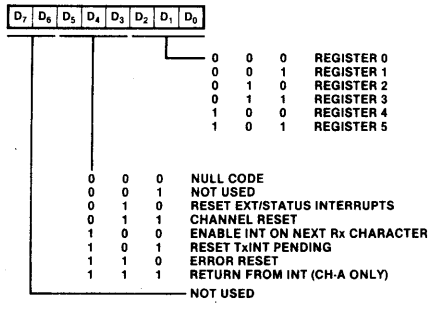
*Used With Special Receive Condition Mode

READ REGISTER 2

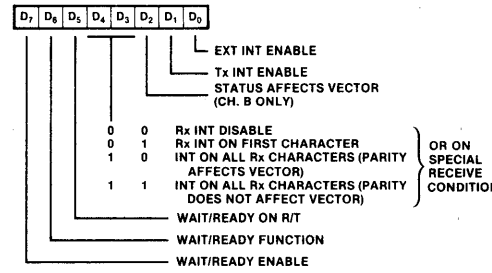


**Variable If "Status Affects Vector" Is Programmed

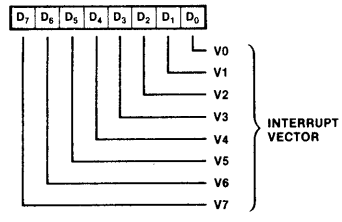
WRITE REGISTER 0



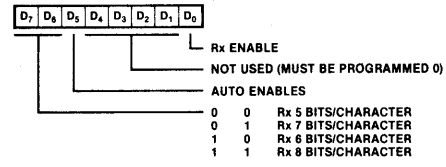
WRITE REGISTER 1



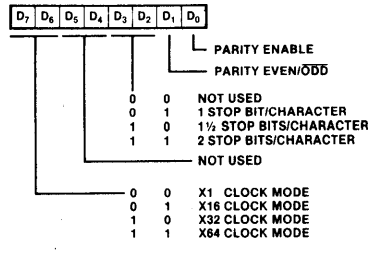
WRITE REGISTER 2 (CHANNEL B ONLY)



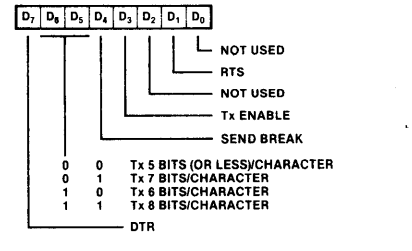
WRITE REGISTER 3



WRITE REGISTER 4



WRITE REGISTER 5



Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

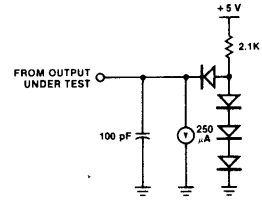
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

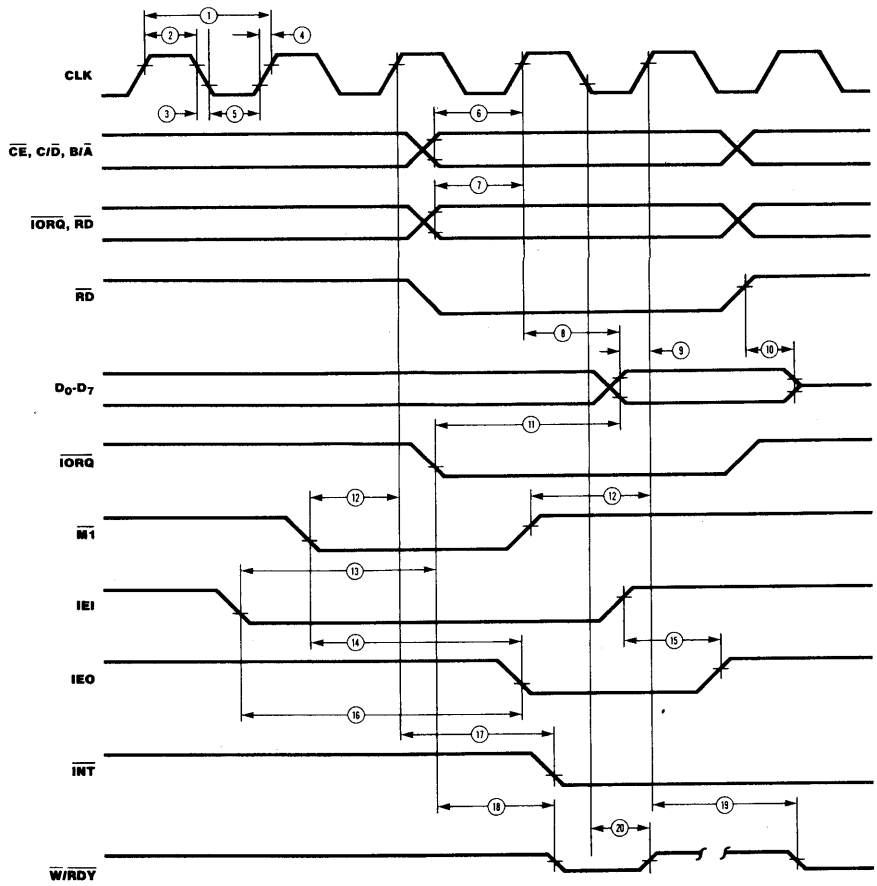


DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	+5.5	V	
V_{IL}	Input Low Voltage	-0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4		V	I_{OH} = -250 µA
I_L	Input/3-State Output Leakage Current	-10	+10	µA	0.4<V<2.4V
$I_{L(RI)}$	$\overline{RI}$ Pin Leakage Current	-40	+10	µA	0.4<V<2.4V
I_{CC}	Power Supply Current		100	mA	

T_A = 0°C to 70°C, V_{CC} = +5V, ±5%

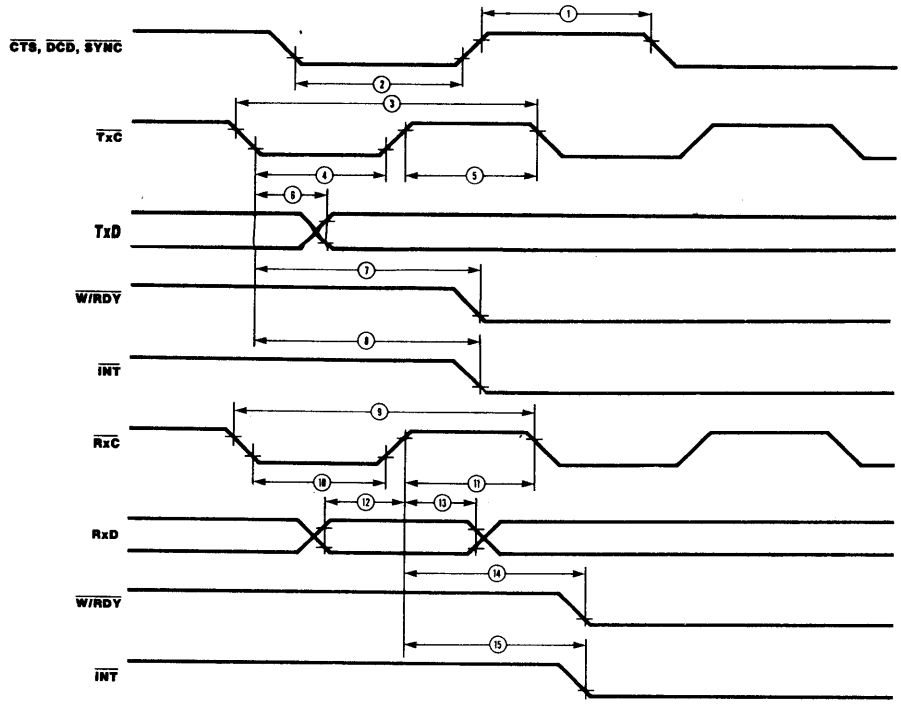
AC Electrical Characteristics



Number	Symbol	Parameter	Z-80 DART Min	Z-80 DART Max	Z-80A DART Min	Z-80A DART Max	Z-80B DART*† Min	Z-80B DART*† Max
1	TcC	Clock Cycle Time	400	4000	250	4000	165	4000
2	TwCh	Clock Width (High)	170	2000	105	2000	70	2000
3	TfC	Clock Fall Time		30		30		15
4	TrC	Clock Rise Time		30		30		15
5	TwCl	Clock Width (Low)	170	2000	105	2000	70	2000
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	160		145		60	
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	240		115		60	
8	TdC(DO)	Clock ↑ to Data Out Delay		240		220		150
9	TsDI(C)	Data In to Clock ↑ Setup Time (Write or M1 Cycle)	50		50		30	
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		230		110		90
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		340		160		100
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	210		90		75	
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	200		140		120	
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before M1)		300		190		160
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		150		100		70
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		150		100		70
17	TdC(INT)	Clock ↑ to INT ↓ Delay		200		200		150
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to W/RDY ↓ Delay (Wait Mode)		300		210		175
19	TdC(W/RR)	Clock ↑ to W/RDY ↓ Delay (Ready Mode)		120		120		100
20	TdC(W/RWz)	Clock ↓ to W/RDY Float Delay (Wait Mode)		150		130		110

*All timings are preliminary and subject to change.
†Units in ns.

AC Electrical Characteristics (Continued)



Number	Symbol	Parameter	Z-80 DART Min	Z-80 DART Max	Z-80A DART Min	Z-80A DART Max	Z-80B DART[1] Min	Z-80B DART[1] Max	Notes†
1	TwPh	Pulse Width (High)	200		200		200		2
2	TwPl	Pulse Width (Low)	200		200		200		2
3	TcTxC	$\overline{TxC}$ Cycle Time	400	∞	400	∞	330	∞	2
4	TwTxCl	$\overline{TxC}$ Width (Low)	180	∞	180	∞	100	∞	2
5	TwTxCh	$\overline{TxC}$ Width (High)	180	∞	180	∞	100	∞	2
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay		400		300		220	2
7	TdTxC(W/RRf)	$\overline{TxC}$ ↓ to W/RDY ↓ Delay (Ready Mode)	5	9	5	9	5	9	3
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	5	9	3
9	TcRxC	$\overline{RxC}$ Cycle Time	400	∞	400	∞	330	∞	2
10	TwRxCl	$\overline{RxC}$ Width (Low)	180	∞	180	∞	100	∞	2
11	TwRxCh	$\overline{RxC}$ Width (High)	180	∞	180	∞	100	∞	2
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		0		2
13	ThRxD(RxC)	RxD Hold Time (x1 Mode)	140		140		100		2
14	TdRxC(W/RRf)	$\overline{RxC}$ ↑ to W/RDY ↓ Delay (Ready Mode)	10	13	10	13	10	13	3
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↓ Delay	10	13	10	13	10	13	3

NOTES:

† In all modes, the System Clock rate must be at least five times the maximum data rate. $\overline{RESET}$ must be active a minimum of one complete clock cycle.

1. Timings are preliminary and subject to change.
2. Units in nanoseconds (ns).
3. Units equal to System Clock Periods.

Product Number	Package/ Temp	Speed	Description
Z8470	CE	2.5 MHz	Z80 DART (40-pin)
Z8470	CM	2.5 MHz	Same as above
Z8470	CMB	2.5 MHz	Same as above
Z8470	CS	2.5 MHz	Same as above
Z8470	DE.	2.5 MHz	Same as above
Z8470	DS	2.5 MHz	Same as above
Z8470	PE	2.5 MHz	Same as above
Z8470	PS	2.5 MHz	Same as above
Z8470A	CE	4.0 MHz	Z80A DART (40-pin)
Z8470A	CM	4.0 MHz	Same as above
Z8470A	CMB	4.0 MHz	Same as above
Z8470A	CS	4.0 MHz	Z80A DART (40-pin)
Z8470A	DE	4.0 MHz	Same as above
Z8470A	DS	4.0 MHz	Same as above
Z8470A	PE	4.0 MHz	Same as above
Z8470A	PS	4.0 MHz	Same as above
Z8470B	CE	6.0 MHz	Z80B DART (40-pin)
Z8470B	CS	6.0 MHz	Same as above
Z8470B	DS	6.0 MHz	Same as above
Z8470B	PS	6.0 MHz	Same as above

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z80L

Family

Zilog

Pioneering the Microworld

Z8300 Low Power Z80L® CPU Central Processing Unit

Zilog

Product Specification

September 1983

Features

- The Z80L combines the high performance of the Z80 CPU with extremely low power consumption. It has the identical pinout and instruction set of the Z80. The result is increased reliability and lower system power requirements. This dramatic power savings makes the Z80L a natural choice for both hand-held and battery backup applications.
- The Z80L CPU is offered in two versions:
  Z8300-1—1.0 MHz clock, 15 mA typical current consumption
  Z8300-3—2.5 MHz clock, 25 mA typical current consumption
- The extensive instruction set contains 158 instructions, resulting in sophisticated data handling capabilities. The 78 instructions of the 8080A are included as a subset; 8080A and Z80 Family software compatibility is maintained.
- The Z80L microprocessors and associated family of peripheral controllers are linked by a vectored interrupt system. This system can be daisy-chained to allow implementation of a priority interrupt scheme. Little, if any, additional logic is required for daisy-chaining.
- Duplicate sets of both general-purpose and flag registers are provided, easing the design and operation of system software. Two 16-bit index registers facilitate program processing of tables and arrays.
- There are three modes of high-speed interrupt processing: 8080 similar, non-Z80 peripheral device, and Z80 Family peripheral with or without daisy chain.
- On-chip dynamic memory refresh counter.

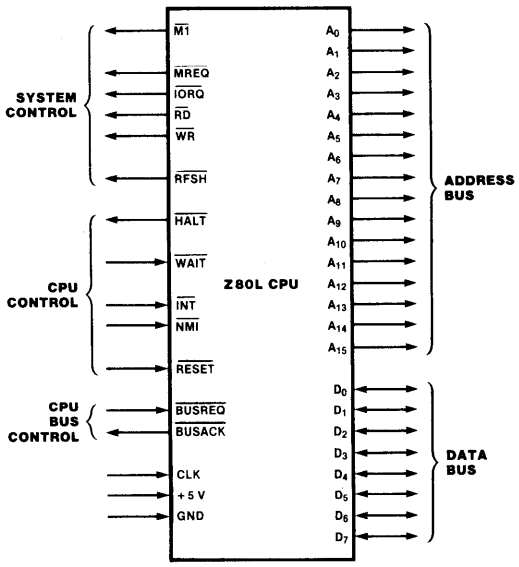


Figure 1. Pin Functions

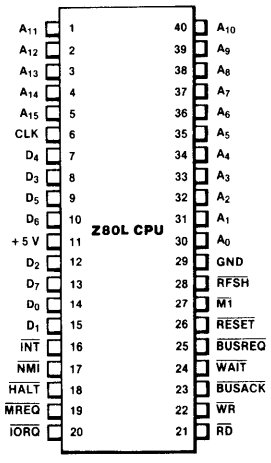


Figure 2. Pin Assignments

General Description

The Z80L CPUs are fourth-generation microprocessors with exceptional computational power. They offer high system throughput and efficient memory utilization combined with extremely low power consumption. The internal registers contain 208 bits of read/write memory that are accessible to the programmer. These registers include two sets of six general-purpose registers which may be used individually as either 8-bit registers or as 16-bit register pairs. In addition, there are two sets of accumulator and flag registers. A group of "Exchange" instructions makes either set of main or alternate registers accessible to the programmer. The alternate set allows operation in foreground-background mode or it may be reserved for very fast interrupt response.

The Z80L also contains a Stack Pointer, Program Counter, two index registers, a Refresh register (counter), and an Interrupt register. The CPU is easy to incorporate into a system since it requires only a single +5 V power source, all output signals are fully decoded and timed to control standard memory or peripheral circuits, and it is supported by an extensive family of peripheral controllers. The internal block diagram (Figure 3) shows the primary functions of the Z80L processors. Subsequent text provides more detail on the Z80L I/O controller family, registers, instruction set, interrupts and daisy chaining, CPU timing, and low power requirements.

Z80L Low Power Feature. The Z80L Family offers state-of-the-art microprocessor performance with extremely low power consumption. Its low power requirement rivals comparable CMOS microprocessors. The Z80L Family's lower power consumption provides the ability to reduce system power requirements and enables its use in applications not previously possible. The Z80L is very well suited to battery backup applications or to systems operating primarily on batteries in hand-held or portable systems.

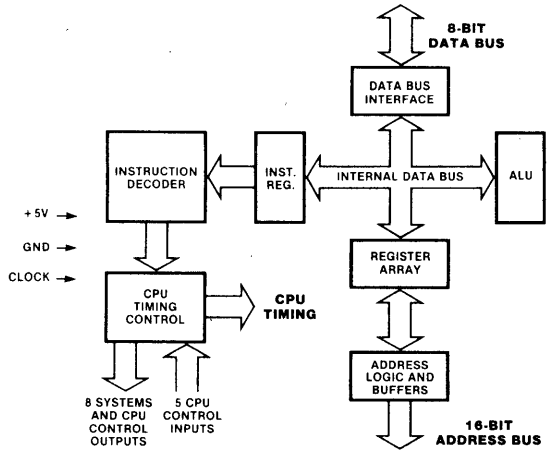


Figure 3. Z80L CPU Block Diagram

Z80L Microprocessor Family

The Zilog Z80L microprocessor is the central element of a comprehensive microprocessor product family. This family works together in most applications with minimum requirements for additional logic, facilitating the design of efficient and cost-effective microcomputer-based systems.

The Z80 Family components provide extensive support for the Z80L microprocessor. These are:

- The PIO (Parallel Input/Output) operates in both data-byte I/O transfer mode (with handshaking) and in bit mode (without handshaking). The PIO may be configured to interface with standard parallel peripheral devices such as printers, tape punches, and keyboards.
- The CTC (Counter/Timer Circuit) features four programmable 8-bit counter/timers, each of which has an 8-bit prescaler. Each of the four channels may be configured to operate in either counter or timer mode.
- The DMA (Direct Memory Access) controller provides dual port data transfer operations and the ability to terminate data transfer as a result of a pattern match.
- The SIO (Serial Input/Output) controller offers two channels. It is capable of operating in a variety of programmable modes for both synchronous and asynchronous communication, including Bi-Synch and SDLC.
- The DART (Dual Asynchronous Receiver/Transmitter) device provides low cost asynchronous serial communication. It has two channels and a full modem control interface.
- These peripherals are also available in a low power version with the exception of the DMA.

Z80L CPU Registers

Figure 4 shows three groups of registers within the Z80L CPU. The first group consists of duplicate sets of 8-bit registers: a principal set and an alternate set (designated by ' [prime], e.g., A'). Both sets consist of the Accumulator Register, the Flag Register, and six general-purpose registers. Transfer of data between these duplicate sets of registers is accomplished by use of "Exchange" instructions. The result is faster response to interrupts and easy, efficient implementation of such versatile programming techniques as background-foreground data processing. The second set of registers consists of six registers with assigned functions. These are the I (Interrupt Register), the R (Refresh Register), the IX and IY (Index Registers), the SP (Stack Pointer), and the PC (Program Counter). The third group consists of two interrupt status flip-flops, plus an additional pair of flip-flops which assists in identifying the interrupt mode at any particular time. Table 1 provides further information on these registers.

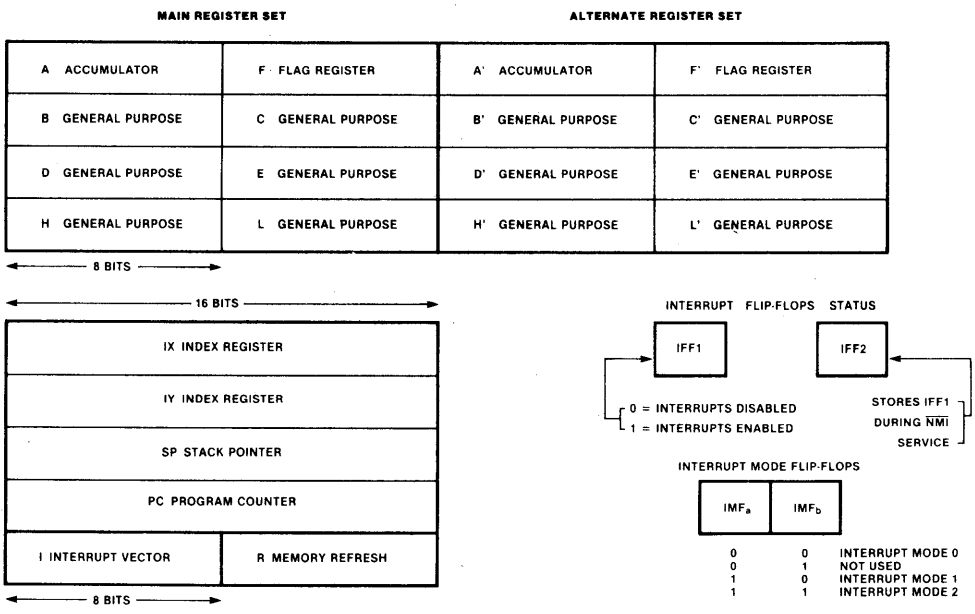


Figure 4. CPU Registers

Z80L CPU Registers (Continued)

Register		Size (Bits)	Remarks
A, A'	Accumulator	8	Stores an operand or the results of an operation.
F, F'	Flags	8	See Instruction Set.
B, B'	General Purpose	8	Can be used separately or as a 16-bit register with C.
C, C'	General Purpose	8	See B, above.
D, D'	General Purpose	8	Can be used separately or as a 16-bit register with E.
E, E'	General Purpose	8	See D, above.
H, H'	General Purpose	8	Can be used separately or as a 16-bit register with L.
L, L'	General Purpose	8	See H, above.
			Note: The (B,C), (D,E), and (H,L) sets are combined as follows: B — High byte C — Low byte D — High byte E — Low byte H — High byte L — Low byte
I	Interrupt Register	8	Stores upper eight bits of memory address for vectored interrupt processing.
R	Refresh Register	8	Provides user-transparent dynamic memory refresh. Lower seven bits are automatically incremented and all eight are placed on the address bus during each instruction fetch cycle refresh time.
IX	Index Register	16	Used for indexed addressing.
IY	Index Register	16	Same as IX, above.
SP	Stack Pointer	16	Holds address of the top of the stack. See Push or Pop in instruction set.
PC	Program Counter	16	Holds address of next instruction.
IFF_1-IFF_2	Interrupt Enable	Flip-Flops	Set or reset to indicate interrupt status (see Figure 4).
IMFa-IMFb	Interrupt Mode	Flip-Flops	Reflect Interrupt mode (see Figure 4).

Table 1. Z80L CPU Registers

Interrupts: General Operation

The CPU accepts two interrupt input signals: $\overline{NMI}$ and $\overline{INT}$. The NMI is a non-maskable interrupt and has the highest priority. $\overline{INT}$ is a lower priority interrupt and it requires that interrupts be enabled in software in order to operate. $\overline{INT}$ can be connected to multiple peripheral devices in a wired-OR configuration.

The Z80L has a single response mode for interrupt service for the non-maskable interrupt. The maskable interrupt, $\overline{INT}$, has three programmable response modes available. These are:

- Mode 0 — similar to the 8080 microprocessor.
- Mode 1 — Peripheral Interrupt service, for use with non-8080/Z80 systems.
- Mode 2 — a vectored interrupt scheme, usually daisy-chained, for use with Z80 Family and compatible peripheral devices.

The CPU services interrupts by sampling the $\overline{NMI}$ and $\overline{INT}$ signals at the rising edge of the last clock of an instruction. Further interrupt service processing depends upon the type of interrupt that was detected. Details on interrupt responses are shown in the CPU Timing Section.

Interrupts: General Operation (Continued)

Non-Maskable Interrupt ($\overline{NMI}$). The non-maskable interrupt cannot be disabled by program control and therefore will be accepted all times by the CPU. $\overline{NMI}$ is usually reserved for servicing only the highest priority type interrupts, such as that for orderly shutdown after power failure has been detected. After recognition of the $\overline{NMI}$ signal (providing $\overline{BUSREQ}$ is not active), the CPU jumps to restart location 0066H. Normally, software starting at this address contains the interrupt service routine.

Maskable Interrupt ($\overline{INT}$). Regardless of the interrupt mode set by the user, the Z80L response to a maskable interrupt input follows a common timing cycle. After the interrupt has been detected by the CPU (provided that interrupts are enabled and $\overline{BUSREQ}$ is not active) a special interrupt processing cycle begins. This is a special fetch ($\overline{M1}$) cycle in which $\overline{IORQ}$ becomes active rather than $\overline{MREQ}$, as in a normal $\overline{M1}$ cycle. In addition, this special $\overline{M1}$ cycle is automatically extended by two $\overline{WAIT}$ states, to allow for the time required to acknowledge the interrupt request and to place the interrupt vector on the bus.

Mode 0 Interrupt Operation. This mode is similar to the 8080 microprocessor interrupt service procedures. The interrupting device places an instruction on the data bus. This is normally a Restart instruction, which will initiate a call to the selected one of eight restart locations in page zero of memory. Unlike the 8080, the Z80 CPU responds to the Call instruction with only one interrupt acknowledge cycle followed by two memory read cycles.

Mode 1 Interrupt Operation. Mode 1 operation is very similar to that for the $\overline{NMI}$. The principal difference is that the Mode 1 interrupt has a vector address of 0038H only.

Mode 2 Interrupt Operation. This interrupt mode has been designed to utilize most effectively the capabilities of the Z80L microprocessor and its associated peripheral family. The interrupting peripheral device selects the starting address of the interrupt service routine. It does this by placing an 8-bit vector on the data bus during the interrupt acknowledge cycle. The CPU forms a pointer using this byte as the lower 8 bits and the contents of the I register as the upper 8 bits. This points to an entry in a table of addresses for interrupt service routines. The CPU then calls the routine at that address. This flexibility in selecting the interrupt service routine address allows the peripheral device to use several different types of service routines. These routines may be located at any available location in memory. Since the interrupting device supplies the low-order byte of the 2-byte vector, bit 0 (A_0) must be a zero.

Interrupt Priority (Daisy Chaining and Nested Interrupts). The interrupt priority of each peripheral device is determined by its physical location within a daisy-chain configuration. Each device in the chain has an interrupt enable input line (IEI) and an interrupt enable output line (IEO), which is fed to the next lower priority device. The first device in the daisy chain has its IEI input hardwired to a High level. The first device has highest priority, while each succeeding device has a corresponding lower priority. This arrangement permits the CPU to select the highest priority interrupt from several simultaneously interrupting peripherals.

The interrupting device disables its IEO line to the next lower priority peripheral until it has been serviced. After servicing, its IEO line is raised, allowing lower priority peripherals to demand interrupt servicing.

The Z80L CPU will nest (queue) any pending interrupts or interrupts received while a selected peripheral is being serviced.

Interrupt Enable/Disable Operation. Two flip-flops, IFF_1 and IFF_2, referred to in the register description are used to signal the CPU interrupt status. Operation of the two flip-flops is described in Table 2. For more details, refer to the *Z80 CPU Technical Manual* and *Z80 Assembly Language Manual.*

Action	IFF_1	IFF_2	Comments
CPU Reset	0	0	Maskable interrupt $\overline{INT}$ disabled
DI instruction execution	0	0	Maskable interrupt $\overline{INT}$ disabled
EI instruction execution	1	1	Maskable interrupt $\overline{INT}$ enabled
LD A,I instruction execution	•	•	$IFF_2 \rightarrow$ Parity flag
LD A,R instruction execution	•	•	$IFF_2 \rightarrow$ Parity flag
Accept $\overline{NMI}$	0	IFF_1	$IFF_1 \rightarrow IFF_2$ (Maskable interrupt $\overline{INT}$ disabled)
RETN instruction execution	IFF_2	•	$IFF_2 \rightarrow IFF_1$ at completion of an $\overline{NMI}$ service routine.

Table 2. State of Flip-Flops

Instruction Set

The Z80L microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor and identical to that of the Z80. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80L instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-XX) and *Assembly Language Programming Manual* (03-0002-XX) contain significantly more details for programming use.

The instructions in Table 2 are divided into the following categories:

- ☐ 8-bit loads
- ☐ 16-bit loads
- ☐ Exchanges, block transfers, and searches
- ☐ 8-bit arithmetic and logic operations
- ☐ General-purpose arithmetic and CPU control
- ☐ 16-bit arithmetic operations
- ☐ Rotates and shifts
- ☐ Bit set, reset, and test operations
- ☐ Jumps
- ☐ Calls, returns, and restarts
- ☐ Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- ☐ Immediate
- ☐ Immediate extended
- ☐ Modified page zero
- ☐ Relative
- ☐ Extended
- ☐ Indexed
- ☐ Register
- ☐ Register indirect
- ☐ Implied
- ☐ Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg.
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110		2	2	7	000 B
								← n →							001 C
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	010 D
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	011 E
										01 r 101					100 H
										← d →					101 L
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	111 A
										01 r 110					
										← d →					
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101	DD	3	5	19	
										01 110 r					
										← d →					
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101	FD	3	5	19	
										01 110 r					
										← d →					
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110	36	2	3	10	
										← n →					
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101	DD	4	5	19	
										00 110 110	36				
										← d →					
										← n →					
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101	FD	4	5	19	
										00 110 110	36				
										← d →					
										← n →					
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010	3A	3	4	13	
										← n →					
										← n →					
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010	32	3	4	13	
										← n →					
										← n →					
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 010 111	57				
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101	ED	2	2	9	
										01 011 111	5F				
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 000 111	47				
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101	ED	2	2	9	
										01 001 111	4F				

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.
IFF the content of the interrupt enable flip flop, (IFF) is copied into the P/V flag.
For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables.

16-Bit Load Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LD dd, nn	dd ← nn	•	•	X	X	•	•	00 dd0 001 ← n → ← n →	3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	X	•	•	11 011 101 DD 00 100 001 21 ← n → ← n →	4	4	14	
LD IY, nn	IY ← nn	•	•	X	X	•	•	11 111 101 FD 00 100 001 21 ← n → ← n →	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	X	X	•	•	00 101 010 2A ← n → ← n →.	3	5	16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	•	•	X	X	•	•	11 101 101 ED 01 dd1 011 ← n → ← n →	4	6	20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	•	•	X	X	•	•	11 011 101 DD 00 101 010 2A ← n → ← n →	4	6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	X	X	•	•	11 111 101 FD 00 101 010 2A ← n → ← n →	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	X	X	•	•	00 100 010 22 ← n → ← n →	3	5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	X	X	•	•	11 101 101 ED 01 dd0 011 ← n → ← n →	4	6	20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	X	X	•	•	11 011 101 DD 00 100 010 22 ← n → ← n →	4	6	20	
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	X	X	•	•	11 111 101 FD 00 100 010 22 ← n → ← n →	4	6	20	
LD SP, HL	SP ← HL	•	•	X	X	•	•	11 111 001 F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	X	•	•	11 011 101 DD 11 111 001 F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	X	•	•	11 111 101 FD 11 111 001 F9	2	2	10	
PUSH qq	(SP − 2) ← qq_L (SP − 1) ← qq_H SP → SP − 2	•	•	X	X	•	•	11 qq0 101	1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP − 2) ← IX_L (SP − 1) ← IX_H SP → SP − 2	•	•	X	X	•	•	11 011 101 DD 11 100 101 E5	2	4	15	
PUSH IY	(SP − 2) ← IY_L (SP − 1) ← IY_H SP → SP − 2	•	•	X	X	•	•	11 111 101 FD 11 100 101 E5	2	4	15	
POP qq	qq_H ← (SP + 1) qq_L ← (SP) SP → SP + 2	•	•	X	X	•	•	11 qq0 001	1	3	10	
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP → SP + 2	•	•	X	X	•	•	11 011 101 DD 11 100 001 E1	2	4	14	
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP → SP + 2	•	•	X	X	•	•	11 111 101 FD 11 100 001 E1	2	4	14	

NOTES: dd is any of the register pairs BC, DE, HL, SP.
qq is any of the register pairs AF, BC, DE, HL.
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
e.g., BC_L = C, AF_H = A.

Exchange, Block Transfer, Block Search Groups

EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11 101 011 EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X	•	•	•	00 001 000 08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X	•	•	•	11 011 001 D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	•	•	X	•	X	•	•	•	11 100 011 E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 011 101 DD 11 100 011 E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)	•	•	X	•	X	•	•	•	11 111 101 FD 11 100 011 E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1	•	•	X	0	X	① 1	0	•	11 101 101 ED 10 100 000 A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1 Repeat until BC = 0	•	•	X	0	X	① 0	0	•	11 101 101 ED 10 110 000 B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.

Exchange, Block Transfer, Block Search Groups (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
LDD	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1	•	•	X 0 X	↕ ①	0	•	11 101 101 ED 10 101 000 A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1 Repeat until BC = 0	•	•	X 0 X	0 ②	0	•	11 101 101 ED 10 111 000 B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A − (HL) HL ← HL + 1 BC ← BC − 1	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 100 001 A1	2	4	16	
CPIR	A − (HL) HL ← HL + 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 110 001 B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A − (HL) HL ← HL − 1 BC ← BC − 1	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 101 001 A9	2	4	16	
CPDR	A − (HL) HL ← HL − 1 BC ← BC − 1 Repeat until A = (HL) or BC = 0	↕	↕ ③	X ↕ X	↕ ①	1	•	11 101 101 ED 10 111 001 B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES: ① P/V flag is 0 if the result of BC − 1 = 0, otherwise P/V = 1.
② P/V flag is 0 at completion of instruction only.
③ Z flag is 1 if A = (HL), otherwise Z = 0.

8-Bit Arithmetic and Logical Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
ADD A, r	A ← A + r	↕	↕	X ↕ X	V	0	↕	10 [000] r	1	1	4	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X ↕ X	V	0	↕	11 [000] 110 ← n →	2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X ↕ X	V	0	↕	10 [000] 110	1	2	7	
ADD A, (IX+d)	A ← A + (IX+d)	↕	↕	X ↕ X	V	0	↕	11 011 101 DD 10 [000] 110 ← d →	3	5	19	
ADD A, (IY+d)	A ← A + (IY+d)	↕	↕	X ↕ X	V	0	↕	11 111 101 FD 10 [000] 110 ← d →	3	5	19	
ADC A, s	A ← A+s+CY	↕	↕	X ↕ X	V	0	↕	[001]				s is any of r, n, (HL), (IX+d), (IY+d) as shown for ADD instruction. The indicated bits replace the [000] in the ADD set above.
SUB s	A ← A−s	↕	↕	X ↕ X	V	1	↕	[010]				
SBC A, s	A ← A−s−CY	↕	↕	X ↕ X	V	1	↕	[011]				
AND s	A ← A ∧ s	↕	↕	X 1 X	P	0	0	[100]				
OR s	A ← A ∨ s	↕	↕	X 0 X	P	0	0	[110]				
XOR s	A ← A ⊕ s	↕	↕	X 0 X	P	0	0	[101]				
CP s	A−s	↕	↕	X ↕ X	V	1	↕	[111]				
INC r	r ← r + 1	↕	↕	X ↕ X	V	0	•	00 r [100]	1	1	4	
INC (HL)	(HL) ←(HL) + 1	↕	↕	X ↕ X	V	0	•	00 110 [100]	1	3	11	
INC (IX+d)	(IX+d) ← (IX+d) + 1	↕	↕	X ↕ X	V	0	•	11 011 101 DD 00 110 [100] ← d →	3	6	23	
INC (IY+d)	(IY+d) ← (IY+d) + 1	↕	↕	X ↕ X	V	0	•	11 111 101 FD 00 110 [100] ← d →	3	6	23	
DEC m	m ← m − 1	↕	↕	X ↕ X	V	1	•	[101]				m is any of r, (HL), (IX+d), (IY+d) as shown for INC. DEC same format and states as INC. Replace [100] with [101] in opcode.

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands.	↕	↕	X	↕	X	P	•	↕	00 100 111 27	1	1	4	Decimal adjust accumulator.
CPL	$A \leftarrow \overline{A}$	•	•	X	1	X	•	1	•	00 101 111 2F	1	1	4	Complement accumulator (one's complement).
NEG	$A \leftarrow 0 - A$	↕	↕	X	↕	X	V	1	↕	11 101 101 ED 01 000 100 44	2	2	8	Negate acc. (two's complement).
CCF	$CY \leftarrow \overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111 3F	1	1	4	Complement carry flag.
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111 37	1	1	4	Set carry flag.
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000 00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110 76	1	1	4	
DI ★	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011 F3	1	1	4	
EI ★	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011 FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 ED 01 000 110 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 ED 01 010 110 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 ED 01 011 110 5E	2	2	8	

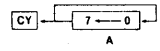
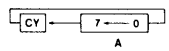
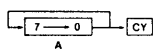
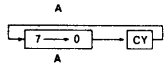
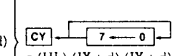
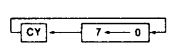
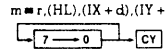
NOTES: IFF indicates the interrupt enable flip-flop.
CY indicates the carry flip-flop.
★ indicates interrupts are not sampled at the end of EI or DI.

16-Bit Arithmetic Group

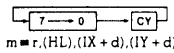
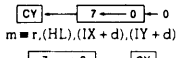
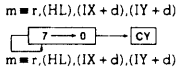
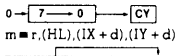
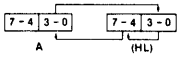
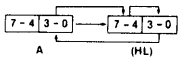
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001	1	3	11	ss Reg. 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 ED 01 ss1 010	2	4	15	
SBC HL, ss	HL ← HL − ss − CY	↕	↕	X	X	X	V	1	↕	11 101 101 ED 01 ss0 010	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 DD 01 pp1 001	2	4	15	pp Reg. 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 FD 00 rr1 001	2	4	15	rr Reg. 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011	1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 100 011 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 100 011 23	2	2	10	
DEC ss	ss ← ss − 1	•	•	X	•	X	•	•	•	00 ss1 011	1	1	6	
DEC IX	IX ← IX − 1	•	•	X	•	X	•	•	•	11 011 101 DD 00 101 011 2B	2	2	10	
DEC IY	IY ← IY − 1	•	•	X	•	X	•	•	•	11 111 101 FD 00 101 011 2B	2	2	10	

NOTES: ss is any of the register pairs BC, DE, HL, SP.
pp is any of the register pairs BC, DE, IX, SP.
rr is any of the register pairs BC, DE, IY, SP.

Rotate and Shift Group

RLCA		•	•	X	0	X	•	0	↕	00 000 111 07	1	1	4	Rotate left circular accumulator.
RLA		•	•	X	0	X	•	0	↕	00 010 111 17	1	1	4	Rotate left accumulator.
RRCA		•	•	X	0	X	•	0	↕	00 001 111 0F	1	1	4	Rotate right circular accumulator.
RRA		•	•	X	0	X	•	0	↕	00 011 111 1F	1	1	4	Rotate right accumulator.
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 r	2	2	8	Rotate left circular register r.
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 CB 00 000 110	2	4	15	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 DD 11 001 011 CB ← d → 00 000 110	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 FD 11 001 011 CB ← d → 00 000 110	4	6	23	
RL m	 m = r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	010				Instruction format and states are as shown for RLC's. To form new opcode replace 000 or RLC's with shown code.
RRC m	 m = r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	001				

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
RR m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	011					
SLA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	100					
SRA m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	101					
SRL m	 m ≡ r,(HL),(IX + d),(IY + d)	↕	↕	X	0	X	P	0	↕	111					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and *right between* the accumulator and location (HL).
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	The content of the upper half of the accumulator is unaffected.

Bit Set, Reset and Test Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
BIT b, r	$Z \leftarrow \overline{r}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX+d)$_b$	$Z \leftarrow \overline{(IX+d)}_b$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 ← d → 01 b 110	DD CB	4	5	20	
BIT b, (IY+d)$_b$	$Z \leftarrow \overline{(IY+d)}_b$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 ← d → 01 b 110	FD CB	4	5	20	
SET b, r	$r_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b r	CB	2	2	8	
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 001 011 11 b 110	CB	2	4	15	
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 011 101 11 001 011 ← d → 11 b 110	DD CB	4	6	23	
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	X	•	X	•	•	•	11 111 101 11 001 011 ← d → 11 b 110	FD CB	4	6	23	
RES b, m	$m_b \leftarrow 0$ m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	10					To form new opcode replace 11 of SET b, s with 10. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) or location m.

Jump Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 ← n → ← n →	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 ← n → ← n →		3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC+e	•	•	X	•	X	•	•	•	00 011 000 ← e−2 →	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 ← e−2 →	38	2	2	7	If condition not met.
	If C = 1, PC ← PC+e											2	3	12	If condition is met.
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 ← e−2 →	30	2	2	7	If condition not met.
	If C = 0, PC ← PC+e											2	3	12	If condition is met.
JP Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 ← e−2 →	28	2	2	7	If condition not met.
	If Z = 1, PC ← PC+e											2	3	12	If condition is met.
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 ← e−2 →	20	2	2	7	If condition not met.
	If Z = 0, PC ← PC+e											2	3	12	If condition is met.
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
JP (IY)	PC ← IY	•	•	X	X	•	•	•	11 111 101 FD 11 101 001 E9	2	2	8	
DJNZ, e	B ← B − 1 If B = 0, continue	•	•	X	X	•	•	•	00 010 000 10 ← e − 2 →	2	2	8	If B = 0.
	If B ≠ 0, PC ← PC + e									2	3	13	If B ≠ 0.

NOTES: e represents the extension in the relative addressing mode.
e is a signed two's complement number in the range < −126, 129 >.
e − 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e.

Call and Return Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
CALL nn	(SP − 1) ← PC_H (SP − 2) ← PC_L PC ← nn	•	•	X	X	•	•	•	11 001 101 CD ← n → ← n →	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	X	•	•	•	11 cc 100 ← n → ← n →	3	3	10	If cc is false.
										3	5	17	If cc is true.
RET	PC_L ← (SP) PC_H ← (SP + 1)	•	•	X	X	•	•	•	11 001 001 C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	X	•	•	•	11 cc 000	1	1	5	If cc is false.
										1	3	11	If cc is true.
RETI	Return from interrupt	•	•	X	X	•	•	•	11 101 101 ED 01 001 101 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	X	•	•	•	11 101 101 ED 01 000 101 45	2	4	14	
RST p	(SP − 1) ← PC_H (SP − 2) ← PC_L PC_H ← 0 PC_L ← p	•	•	X	X	•	•	•	11 t 111	1	3	11	

cc Condition: 000 NZ non-zero; 001 Z zero; 010 NC non-carry; 011 C carry; 100 PO parity odd; 101 PE parity even; 110 P sign positive; 111 M sign negative

t p: 000 00H; 001 08H; 010 10H; 011 18H; 100 20H; 101 28H; 110 30H; 111 38H

NOTE: [1]RETN loads IFF_2 → IFF_1

Input and Output Group

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
IN A, (n)	A ← (n)	•	•	X	•	X	•	•	11 011 011 DB ← n →	2	3	11	n to A_0 ~ A_7 Acc. to A_8 ~ A_{15}
IN r, (C)	r ← (C) if r = 110 only the flags will be affected	↕	↕ ①	X	P	0	•	11 101 101 ED 01 r 000	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INI	(HL) ← (C) B ← B − 1 HL ← HL + 1	X	↕ ②	X	X	X	1	X	11 101 101 ED 10 100 010 A2	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INIR	(HL) ← (C) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1 ②	X	X	X	1	X	11 101 101 ED 10 110 010 B2	2	5 (If B ≠ 0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
										2	4 (If B = 0)	16	
IND	(HL) ← (C) B ← B − 1 HL ← HL − 1	X	↕ ②	X	X	X	1	X	11 101 101 ED 10 101 010 AA	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
INDR	(HL) ← (C) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1 ②	X	X	X	1	X	11 101 101 ED 10 111 010 BA	2	5 (If B ≠ 0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
										2	4 (If B = 0)	16	
OUT (n), A	(n) ← A	•	•	X	•	X	•	•	11 010 011 D3 ← n →	2	3	11	n to A_0 ~ A_7 Acc. to A_8 ~ A_{15}
OUT (C), r	(C) ← r	•	•	X	•	X	•	•	11 101 101 ED 01 r 001	2	3	12	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OUTI	(C) ← (HL) B ← B − 1 HL ← HL + 1	X	↕ ②	X	X	X	1	X	11 101 101 ED 10 100 011 A3	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}
OTIR	(C) ← (HL) B ← B − 1 HL ← HL + 1 Repeat until B = 0	X	1 ②	X	X	X	1	X	11 101 101 ED 10 110 011 B3	2	5 (If B ≠ 0)	21	C to A_0 ~ A_7 B to A_8 ~ A_{15}
										2	4 (If B = 0)	16	
OUTD	(C) ← (HL) B ← B − 1 HL ← HL − 1	X	↕ ②	X	X	X	1	X	11 101 101 ED 10 101 011 AB	2	4	16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

NOTE: ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset.
② Z flag is set upon instruction completion only.

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z	H	P/V	N	C	Opcode 76 543 210 Hex	No.of Bytes	No.of M Cycles	No.of T States	Comments
OTDR	(C) ← (HL) B ← B − 1 HL ← HL − 1 Repeat until B = 0	X	1	X	X	X	↕	X	11 101 101 ED 10 111 011	2 2	5 (If B≠0) 4 (If B = 0)	21 16	C to A_0 ~ A_7 B to A_8 ~ A_{15}

Summary of Flag Operation

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry.
SUB s; SBC A, s; CP s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator.
AND s	↕	↕	X	1	X	P	0	0	Logical operations.
OR s, XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment.
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement.
ADD DD, ss	•	•	X	X	X	•	0	↕	16-bit add.
ADC HL, ss	↕	↕	X	X	X	V	0	↕	16-bit add with carry.
SBC HL, ss	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry.
RLA, RLCA, RRA; RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator.
RL m; RLC m; RR m; RRC m; SLA m; SRA m; SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations.
RLD; RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right.
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator.
CPL	•	•	X	1	X	•	1	•	Complement accumulator.
SCF	•	•	X	0	X	•	0	1	Set carry.
CCF	•	•	X	X	X	•	0	↕	Complement carry.
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect.
INI, IND, OUTI; OUTD	X	↕	X	X	X	X	1	•	Block input and output. Z = 0 if B ≠ 0 otherwise Z = 0.
INIR; INDR; OTIR; OTDR	X	1	X	X	X	X	1	•	
LDI; LDD	X	X	X	0	X	↕	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LDIR; LDDR	X	X	X	0	X	0	0	•	
CPI; CPIR; CPD; CPDR	X	↕	X	X	X	↕	1	•	Block search instructions. Z = 1 if A = (HL), otherwise Z = 0. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LD A, I, LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag.

Symbolic Notation

Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ have entered their high-impedance states. The external circuitry can now control these lines.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). Bus Request has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle. $\overline{BUSREQ}$ forces the CPU address bus, data bus, and control signals $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ to go to a high-impedance state so that other devices can control these lines. $\overline{BUSREQ}$ is normally wire-ORed and requires an external pullup for these applications. Extended $\overline{BUSREQ}$ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

$\overline{HALT}$. *Halt State* (output, active Low). $\overline{HALT}$ indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

$\overline{INT}$. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. $\overline{INT}$ is normally wire-ORed and requires an external pullup for these applications.

$\overline{IORQ}$. *Input/Output Request* (output, active Low, 3-state). $\overline{IORQ}$ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. $\overline{IORQ}$ is also generated concurrently with $\overline{M1}$ during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

$\overline{M1}$. *Machine Cycle One* (output, active Low). $\overline{M1}$, together with $\overline{MREQ}$, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. $\overline{M1}$, together with $\overline{IORQ}$, indicates an interrupt acknowledge cycle.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). $\overline{MREQ}$ indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{NMI}$. *Non-Maskable Interrupt* (input, negative edge-triggered). $\overline{NMI}$ has a higher priority than $\overline{INT}$. $\overline{NMI}$ is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

$\overline{RD}$. *Read* (output, active Low, 3-state). $\overline{RD}$ indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that $\overline{RESET}$ must be active for a minimum of three full clock cycles before the reset operation is complete.

$\overline{RFSH}$. *Refresh* (output, active Low). $\overline{RFSH}$, together with $\overline{MREQ}$, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

$\overline{WAIT}$. *Wait* (input, active Low). $\overline{WAIT}$ indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended $\overline{WAIT}$ periods can prevent the CPU from refreshing dynamic memory properly.

$\overline{WR}$. *Write* (output, active Low, 3-state). $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

CPU Timing

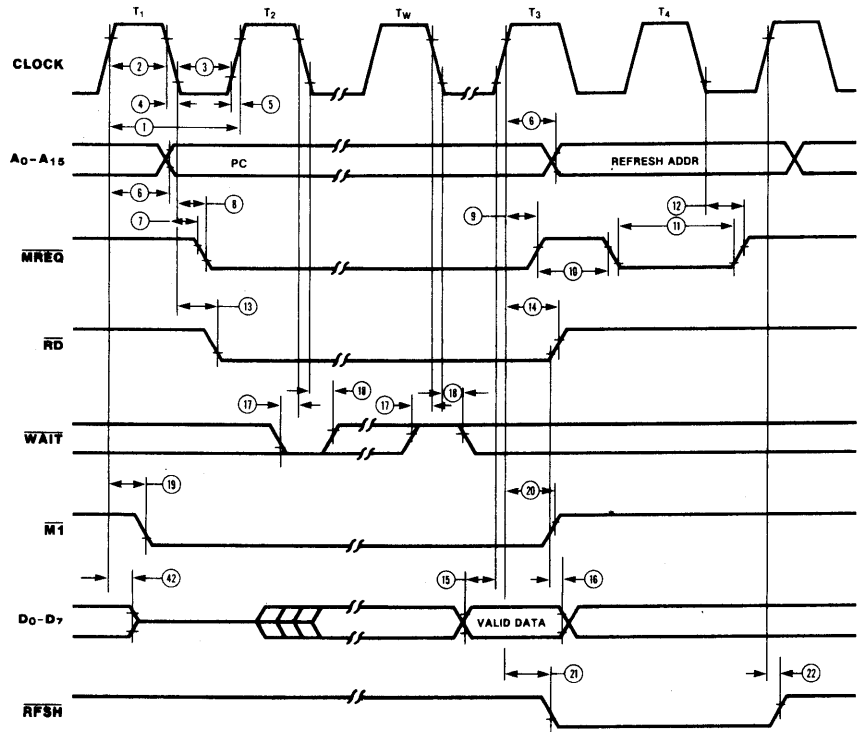
The CPU executes instructions by proceeding through a specific sequence of operations:

- Memory read or write
- I/O device read or write
- Interrupt acknowledge

The basic clock period is referred to as a T time or cycle, and three or more T cycles make up a machine cycle (M1, M2 or M3 for instance). Machine cycles can be extended either by the CPU automatically inserting one or more Wait states or by the insertion of one or more Wait states by the user.

Instruction Opcode Fetch. The CPU places the contents of the Program Counter (PC) on the address bus at the start of the cycle (Figure 5). Approximately one-half clock cycle later, $\overline{MREQ}$ goes active. When active, $\overline{RD}$ indicates that the memory data can be enabled onto the CPU data bus.

The CPU samples the $\overline{WAIT}$ input with the falling edge of clock state T_2. During clock states T_3 and T_4 of an M1 cycle dynamic RAM refresh can occur while the CPU starts decoding and executing the instruction. When the Refresh Control signal becomes active, refreshing of dynamic memory can take place.



NOTE: T_W-Wait cycle added when necessary for slow ancilliary devices.

Figure 5. Instruction Opcode Fetch

CPU Timing (Continued)

Memory Read or Write Cycles. Figure 6 shows the timing of memory read or write cycles other than an opcode fetch ($\overline{M1}$) cycle. The $\overline{MREQ}$ and $\overline{RD}$ signals function exactly as in the fetch cycle. In a memory write cycle, $\overline{MREQ}$ also becomes active when the address bus is stable. The $\overline{WR}$ line is active when the data bus is stable, so that it can be used directly as an R/$\overline{W}$ pulse to most semiconductor memories.

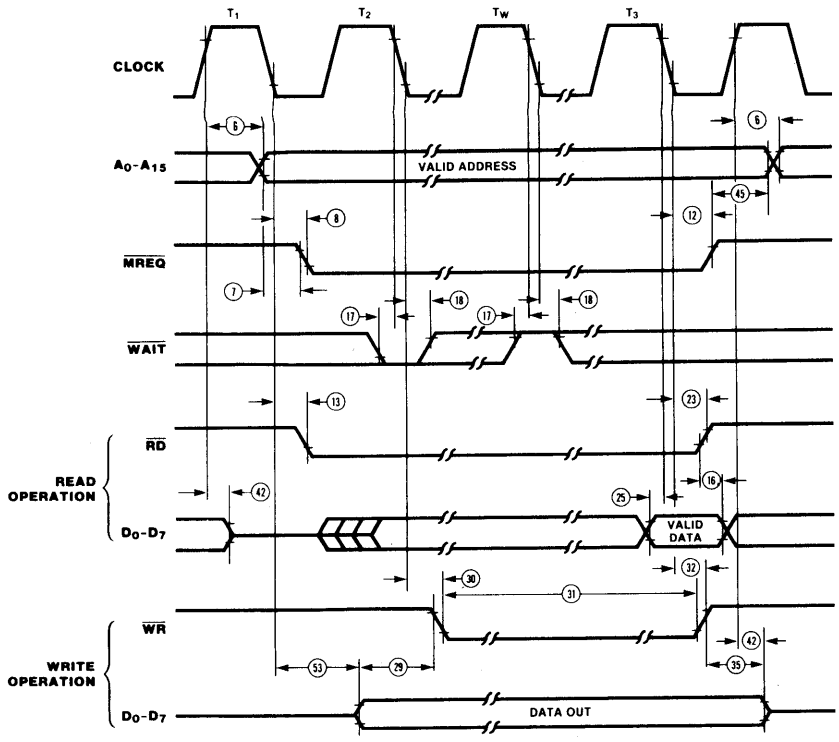
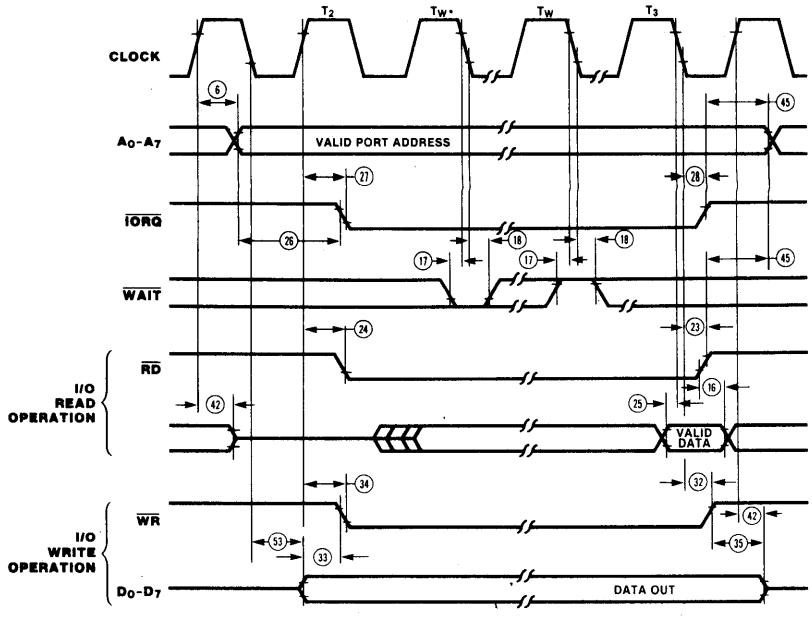


Figure 6. Memory Read or Write Cycles

CPU Timing (Continued)

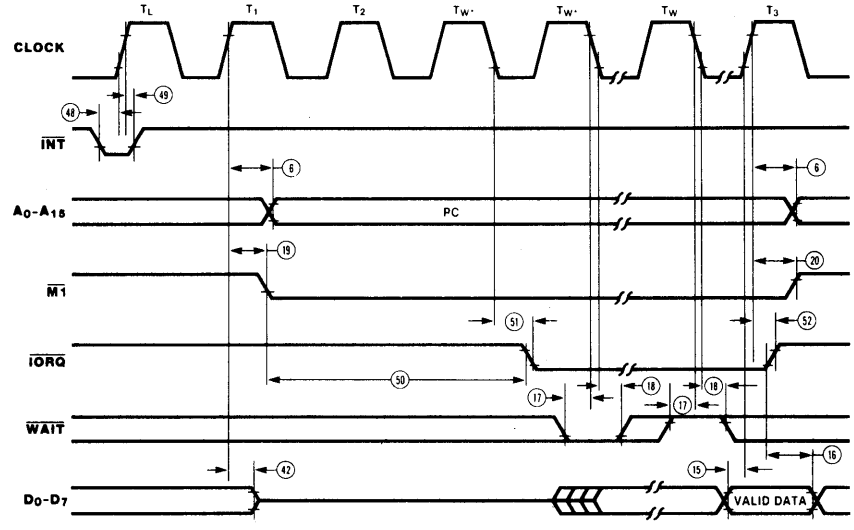
Input or Output Cycles. Figure 7 shows the timing for an I/O read or I/O write operation. During I/O operations, the CPU automatically inserts a single Wait state (T_W). This extra Wait state allows sufficient time for an I/O port to decode the address from the port address lines.



NOTE: T_W* = One Wait cycle automatically inserted by CPU.

Figure 7. Input or Output Cycles

Interrupt Request/Acknowledge Cycle. The CPU samples the interrupt signal with the rising edge of the last clock cycle at the end of any instruction (Figure 8). When an interrupt is accepted, a special $\overline{M1}$ cycle is generated. During this $\overline{M1}$ cycle, $\overline{IORQ}$ becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. The CPU automatically adds two Wait states to this cycle.



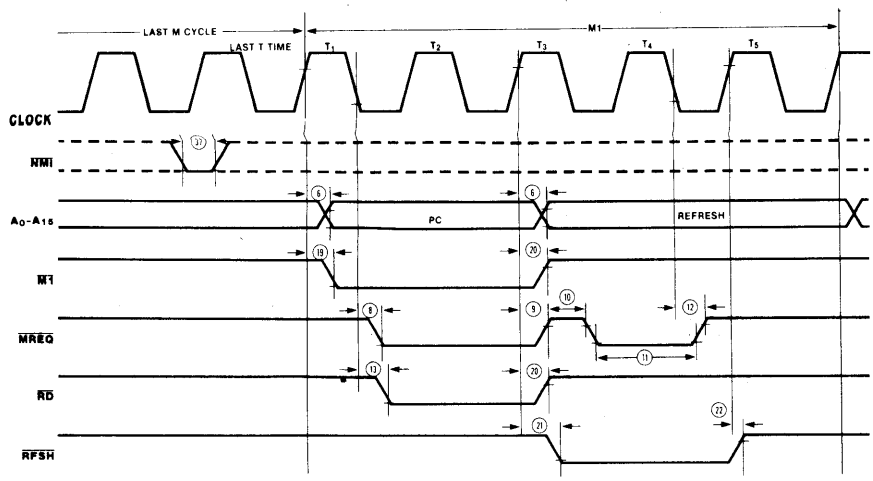
NOTE: 1) T_L = Last state of previous instruction.

2) Two Wait cycles automatically inserted by CPU(*).

Figure 8. Interrupt Request/Acknowledge Cycle

CPU Timing (Continued)

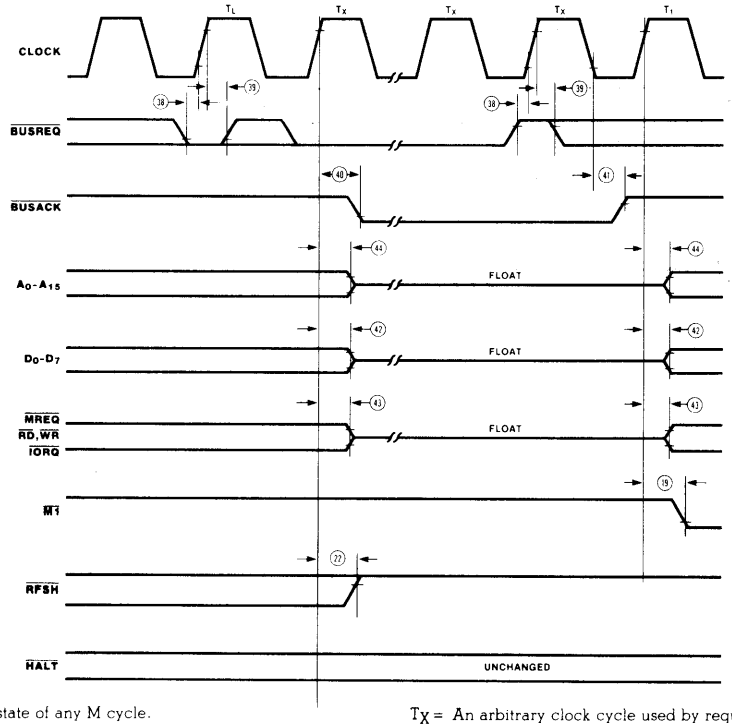
Non-Maskable Interrupt Request Cycle. $\overline{NMI}$ is sampled at the same time as the maskable interrupt input $\overline{INT}$ but has higher priority and cannot be disabled under software control. The subsequent timing is similar to that of a normal memory read operation except that data put on the bus by the memory is ignored. The CPU instead executes a restart (RST) operation and jumps to the $\overline{NMI}$ service routine located at address 0066H (Figure 9).



*Although $\overline{NMI}$ is an asynchronous input, to guarantee its being recognized on the following machine cycle, $\overline{NMI}$'s falling edge must occur no later than the rising edge of the clock cycle preceding T_{LAST}.

Figure 9. Non-Maskable Interrupt Request Operation

Bus Request/Acknowledge Cycle. The CPU samples $\overline{BUSREQ}$ with the rising edge of the last clock period of any machine cycle (Figure 10). If $\overline{BUSREQ}$ is active, the CPU sets its address, data, and $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$ lines to a high-impedance state with the rising edge of the next clock pulse. At that time, any external device can take control of these lines, usually to transfer data between memory and I/O devices.

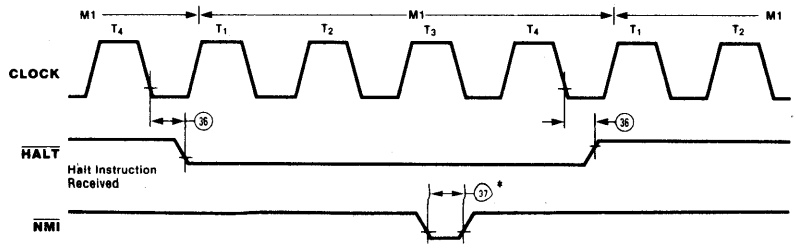


NOTE: T_L = Last state of any M cycle. T_X = An arbitrary clock cycle used by requesting device.

Figure 10. Z-BUS Request/Acknowledge Cycle

CPU Timing (Continued)

Halt Acknowledge Cycle. When the CPU receives a $\overline{HALT}$ instruction, it executes NOP states until either an $\overline{INT}$ or $\overline{NMI}$ input is received. When in the Halt state, the $\overline{HALT}$ output is active and remains so until an interrupt is processed (Figure 11).



NOTE: $\overline{INT}$ will also force a Halt exit.

*See note, Figure 9.

Figure 11. Halt Acknowledge Cycle

Reset Cycle. $\overline{RESET}$ must be active for at least three clock cycles for the CPU to properly accept it. As long as $\overline{RESET}$ remains active, the address and data buses float, and the control outputs are inactive. Once $\overline{RESET}$ goes inactive, two internal T cycles are consumed before the CPU resumes normal processing operation. $\overline{RESET}$ clears the PC register, so the first opcode fetch will be to location 0000 (Figure 12).

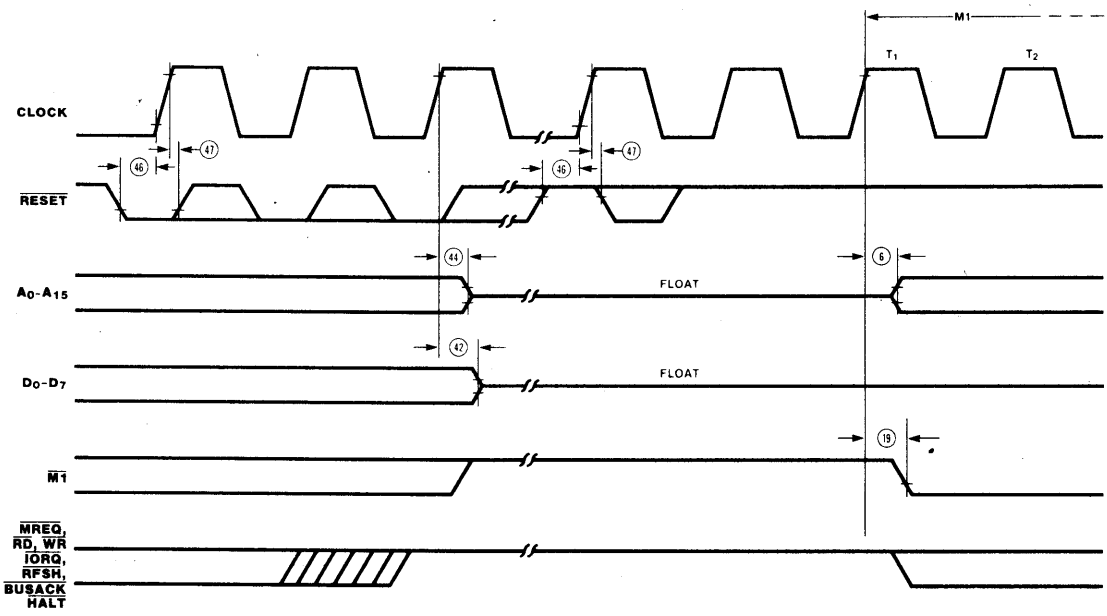


Figure 12. Reset Cycle

AC Characteristics†

Number	Symbol	Parameter	Z8300-1 (1.0 MHz) Min (ns)	Z8300-1 (1.0 MHz) Max (ns)	Z8300-3 (2.5 MHz) Min (ns)	Z8300-3 (2.5 MHz) Max (ns)
1	TcC	Clock Cycle Time	1000*		400*	
2	TwCh	Clock Pulse Width (High)	470*		180*	
3	TwCl	Clock Pulse Width (Low)	470	2000	180	2000
4	TfC	Clock Fall Time	—	30	—	30
5	TrC	Clock Rise Time	—	30	—	30
6	TdCr(A)	Clock ↑ to Address Valid Delay	—	380	—	145
7	TdA(MREQf)	Address Valid to $\overline{MREQ}$ ↓ Delay	370*	—	125*	—
8	TdCf(MREQf)	Clock ↓ to $\overline{MREQ}$ ↓ Delay	—	260	—	100
9	TdCr(MREQr)	Clock ↑ to $\overline{MREQ}$ ↑ Delay	—	260	—	100
10	TwMREQh	$\overline{MREQ}$ Pulse Width (High)	410*		170*	
11	TwMREQl	$\overline{MREQ}$ Pulse Width (Low)	890*	—	360*	—
12	TdCf(MREQr)	Clock ↓ to $\overline{MREQ}$ ↑ Delay	—	260	—	100
13	TdCf(RDf)	Clock ↓ to $\overline{RD}$ ↓ Delay	—	340	—	130
14	TdCr(RDr)	Clock ↑ to $\overline{RD}$ ↑ Delay	—	260	—	100
15	TsD(Cr)	Data Setup Time to Clock ↑	140		50	
16	ThD(RDr)	Data Hold Time to $\overline{RD}$ ↑	—	0	—	0
17	TsWAIT(Cf)	$\overline{WAIT}$ Setup Time to Clock ↓	190	—	70	—
18	ThWAIT(Cf)	$\overline{WAIT}$ Hold Time after Clock ↓	—	0	—	0
19	TdCr(M1f)	Clock ↑ to $\overline{M1}$ ↓ Delay	—	340	—	130
20	TdCr(M1r)	Clock ↑ to $\overline{M1}$ ↑ Delay	—	340	—	130
21	TdCr(RFSHf)	Clock ↑ to $\overline{RFSH}$ ↓ Delay	—	460	—	180
22	TdCr(RFSHr)	Clock ↑ to $\overline{RFSH}$ ↑ Delay	—	390	—	150
23	TdCf(RDr)	Clock ↓ to $\overline{RD}$ ↑ Delay	—	290	—	110
24	TdCr(RDf)	Clock ↑ to $\overline{RD}$ ↓ Delay	—	260	—	100
25	TsD(Cf)	Data Setup to Clock ↓ during M_2, M_3, M_4 or M_5 Cycles	160		60	
26	TdA(IORQf)	Address Stable prior to $\overline{IORQ}$ ↓	790*	—	320*	—
27	TdCr(IORQf)	Clock ↑ to $\overline{IORQ}$ ↓ Delay	—	240	—	90
28	TdCf(IORQr)	Clock ↓ to $\overline{IORQ}$ ↑ Delay	—	290	—	110
29	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	470*	—	190*	—
30	TdCf(WRf)	Clock ↓ to $\overline{WR}$ ↓ Delay	—	240	—	90
31	TwWR	$\overline{WR}$ Pulse Width	890*	—	360*	—
32	TdCf(WRr)	Clock ↓ to $\overline{WR}$ ↑ Delay	—	260	—	100
33	TdD(WRf)	Data Stable prior to $\overline{WR}$ ↓	-30*	—	30*	—
34	TdCr(WRf)	Clock ↑ to $\overline{WR}$ ↓ Delay	—	210	—	80
35	TdWRr(D)	Data Stable from $\overline{WR}$ ↑	290*		130*	
36	TdCf(HALT)	Clock ↓ to $\overline{HALT}$ ↑ or ↓	—	760	—	300
37	TwNMI	$\overline{NMI}$ Pulse Width	210	—	80	—
38	TsBUSREQ(Cr)	$\overline{BUSREQ}$ Setup Time to Clock ↑	210	—	80	—

*For clock periods other than the minimums shown in the table, calculate parameters using the expressions in the table on the following page.
†All timings assume equal loading on pins within 50 pF.
Timings are preliminary and subject to change.

AC Characteristics† (Continued)

Number	Symbol	Parameter	Z8300-1 Min (ns)	Z8300-1 Max (ns)	Z8300-3 Min (ns)	Z8300-3 Max (ns)
39	ThBUSREQ(Cr)	$\overline{BUSREQ}$ Hold Time after Clock ↑	0	—	0	—
40	TdCr(BUSACKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		310		120
41	TdCf(BUSACKr)	Clock ↓ to $\overline{BUSACK}$ ↑ Delay	—	290	—	110
42	TdCr(Dz)	Clock ↑ to Data Float Delay	—	240	—	90
43	TdCr(CTz)	Clock ↑ to Control Outputs Float Delay ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$)	—	290	—	110
44	TdCr(Az)	Clock ↑ to Address Float Delay	—	290	—	110
45	TdCTr(A)	$\overline{MREQ}$ ↑, $\overline{IORQ}$ ↑, $\overline{RD}$ ↑, and $\overline{WR}$ ↑ to Address Hold Time	400*		160*	
46	TsRESET(Cr)	$\overline{RESET}$ to Clock ↑ Setup Time	240	—	90	—
47	ThRESET(Cr)	$\overline{RESET}$ to Clock ↑ Hold Time	—	0	—	0
48	TsINTf(Cr)	$\overline{INT}$ to Clock ↑ Setup Time	210	—	80	—
49	ThINTr(Cr)	$\overline{INT}$ to Clock ↑ Hold Time	—	0	—	0
50	TdM1f(IORQf)	M1 ↓ to $\overline{IORQ}$ ↓ Delay	2300*		920*	
51	TdCf(IORQf)	Clock ↓ to $\overline{IORQ}$ ↓ Delay	—	290	—	110
52	TdCf(IORQr)	Clock ↑ to $\overline{IORQ}$ ↑ Delay	—	260	—	100
53	TdCf(D)	Clock ↓ to Data Valid Delay	—	290	—	230

*For clock periods other than the minimums shown in the table, calculate parameters using the following expressions. Calculated values above assumed TrC = TfC = 20 ns.
† All timings assume equal loading on pins with 50 pF.
Timings are preliminary and subject to change.

Footnotes to AC Characteristics

Number	Symbol	Z8300-1	Z8300-3
1	TcC	TwCh + TwCl + TrC + TfC	TwCh + TwCl + TrC + TfC
2	TwCh	Although static by design, TwCh of greater than 200 μs is not guaranteed	Although static by design, TwCh of greater than 200 μs is not guaranteed
7	TdA(MREQf)	TwCh + TfC − 200	TwCh + TfC − 75
10	TwMREQh	TwCh + TfC − 90	TwCh + TfC − 30
11	TwMREQl	TcC − 110	TcC − 30
26	TdA(IORQf)	TcC − 210	TcC − 80
29	TdD(WRf)	TcC − 540	TcC − 210
31	TwWR	TcC − 110	TcC − 40
33	TdD(WRf)	TwCl + TrC − 470	TwCl + TrC − 180
35	TdWRr(D)	TwCl + TrC − 210	TwCl + TrC − 80
45	TdCTr(A)	TwCl + TrC − 110	TwCl + TrC − 40
50	TdM1f(IORQf)	2TcC + TwCh + TfC − 210	2TcC + TwCh + TfC − 80

AC Test Conditions:
V_{IH} = 2.0 V
V_{IL} = 0.8 V
V_{IHC} = V_{CC} -0.6 V
V_{ILC} = 0.45 V
V_{OH} = 2.0 V
V_{OL} = 0.8 V
FLOAT = ±0.5 V

Absolute Maximum Ratings

Storage Temperature . . . . . -65°C to +150°C
Temperature under Bias . . . . . See Ordering Information
Voltages on all inputs and outputs with respect to ground . -0.3 V to +7 V
Power Dissipation . . . . . 1.5 W

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

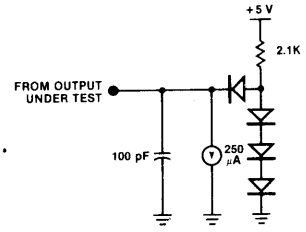
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a load capacitance of 100 pF. Add 10 ns delay for each 50 pF increase in load up to a maximum of 200 pF for the data bus and 100 pF for address and control lines.



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}-.6	V_{CC}+.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 1.8 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current		10	µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage		±10[1]	µA	V_{OUT} = 0.4 to V_{CC}
I_{CC}	Power Supply Current				

	Temperature				
	0°C		25°C	70°C	
Frequency	Max	Max	Typical	Max	Unit
Z8300-1 (1.0 MHz)	30	26	15	20	mA
Z8300-3 (2.5 MHz)	45	42	25	35	mA

1. A_{15}-A_0, D_7-D_0, $\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$, and $\overline{WR}$.

Capacitance

Symbol	Parameter	Min	Max	Unit	Note
C_{CLOCK}	Clock Capacitance		35	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

T_A = 25°C, f = 1 MHz.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8300-1	PS	1.0 MHz	Z80L CPU (40-pin)	Z8300-3	PS	2.5 MHz	Z80L CPU (40-pin)
Z8300-1	CS	1.0 MHz	Same as above	Z8300-3	CS	2.5 MHz	Same as above

NOTES: C = Ceramic, P = Plastic; S = 0°C to +70°C.

Zilog

AC and DC Characteristics

Preliminary

September 1983

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information in Product Specifications

Storage Temperature . . . . . -65°C to +150°C

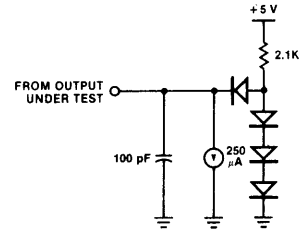
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature range is:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V

*See Ordering Information section in product specifications for package temperature range and product number.



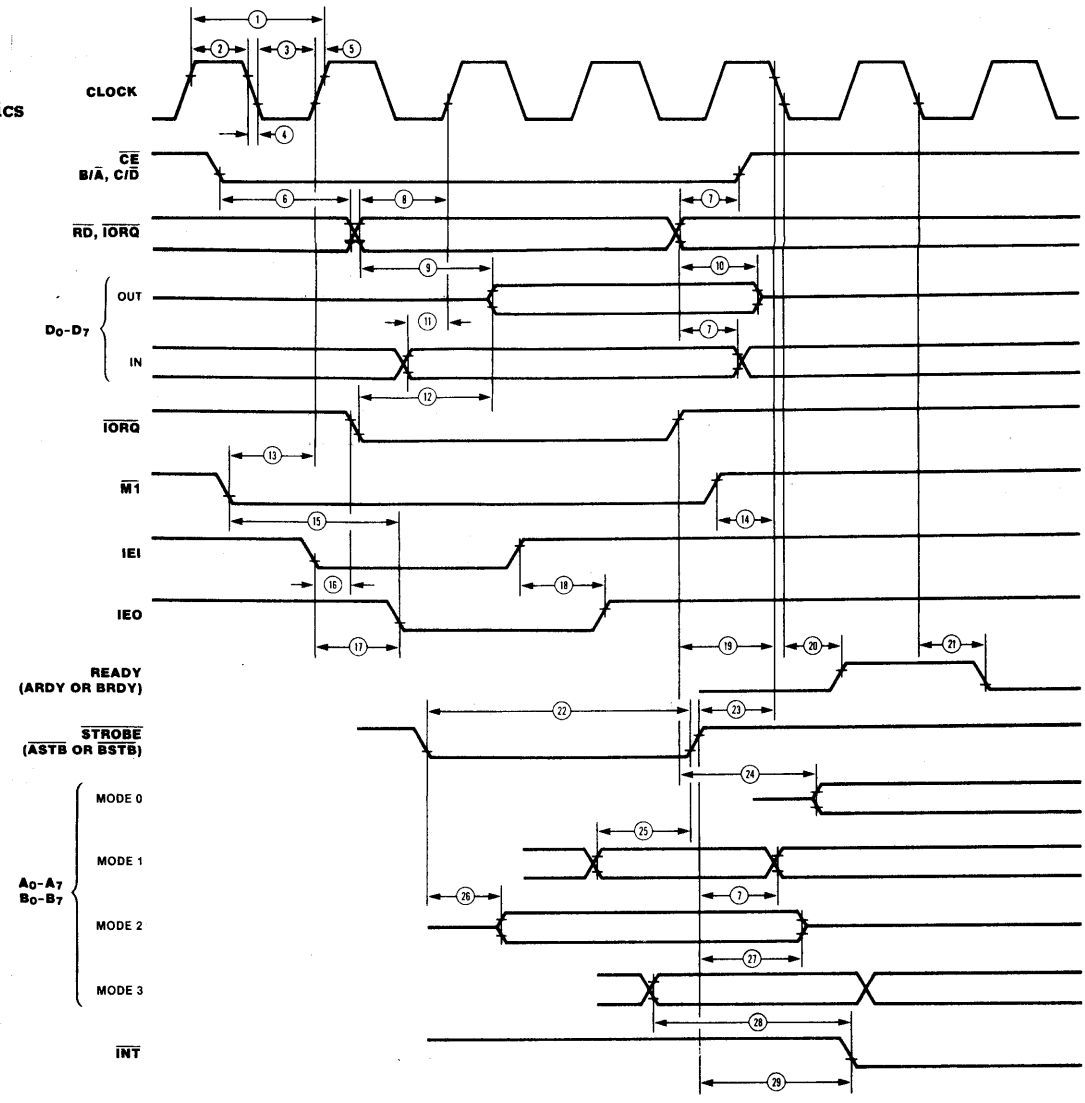
DC Characteristics

Symbol	Parameter	Min	Max	Typical	Unit	Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45		V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	V_{CC}+0.3		V	
V_{IL}	Input Low Voltage	-0.3	+0.8		V	
V_{IH}	Input High Voltage	+2.0	V_{CC}		V	
V_{OL}	Output Low Voltage		+0.4		V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4			V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current		±10		µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10		µA	V_{OUT} = 0.4 to V_{CC}
$I_{L(SY)}$[1]	SYNC Pin Leakage Current		+10/-40		µA	V_{IN} = 0 to V_{CC}
I_{CC}	Power Supply Current:					
	SIO		30	20	mA	
	PIO		20	13	mA	
	CTC		20	13	mA	
I_{OHD}[2]	Darlington Drive Current	-1.5			mA	V_{OH} = 1.5V, R_{EXT} = 390Ω

Over specified temperature and voltage range.

NOTES:
[1] SIO only
[2] CTC and PIO only

Z8320-1 and Z8320-3 Z80L PIO AC Characteristics



Z8320-1 and Z8320-3 Z80L PIO AC Characteristics (Continued)

Number	Symbol	Parameter	Z8320-1 (1.0 MHz) Min	Z8320-1 (1.0 MHz) Max	Z8320-3 (2.5 MHz) Min	Z8320-3 (2.5 MHz) Max	Notes*
1	TcC	Clock Cycle Time	1000		400		[1]
2	TwCH	Clock Width (High)	470	2000	170	2000	
3	TwCl	Clock Width (Low)	470	2000	170	2000	
4	TfC	Clock Fall Time		30		30	
5	TrC	Clock Rise Time		30		30	
6	TsCS(Rl)	$\overline{CE}$, B/$\overline{A}$, C/$\overline{D}$ to $\overline{RD}$, $\overline{IORQ}$ ↓ Setup Time	140		50		[6]
7	Th	Any Hold Times for Specified Setup Time	0		0		
8	TsRI(C)	$\overline{RD}$, $\overline{IORQ}$ to Clock ↑ Setup Time	300		115		
9	TdRI(DO)	$\overline{RD}$, $\overline{IORQ}$ ↓ to Data Out Delay		1090		430	[2]
10	TdRI(DOs)	$\overline{RD}$, $\overline{IORQ}$ ↑ to Data Out Float Delay		410		160	
11	TsDI(C)	Data In to Clock ↑ Setup Time	140		50		CL = 50 pF
12	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		860		340	[3]
13	TsM1(Cr)	$\overline{M1}$ ↓ to Clock ↑ Setup Time	540		210		
14	TsM1(Cf)	$\overline{M1}$ ↑ to Clock ↓ Setup Time (M1 Cycle)	0		0		[8]
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt Immediately Preceding M1 ↓)		760		300	[5, 7]
16	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	360		140		[7]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		480		190	[5] CL = 50 pF
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED Decode)		540		210	[5]
19	TcIO(C)	$\overline{IORQ}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	560		220		
20	TdC(RDYr)	Clock ↓ to READY ↑ Delay		510		200	[5] CL = 50 pF
21	TdC(RDYf)	Clock ↓ to READY ↓ Delay		390		150	[5]
22	TwSTB	$\overline{STROBE}$ Pulse Width	390		150		[4]
23	TsSTB(C)	$\overline{STROBE}$ ↑ to Clock ↓ Setup Time (To Activate READY on Next Clock Cycle)	560		220		[5]
24	TdIO(PD)	$\overline{IORQ}$ ↑ to PORT DATA Stable Delay (Mode 0)		510		200	[5]
25	TsPD(STB)	PORT DATA to $\overline{STROBE}$ ↑ Setup Time (Mode 1)	660		260		
26	TdSTB(PD)	$\overline{STROBE}$ ↓ to PORT DATA Stable (Mode 2)		590		230	[5]
27	TdSTB(PDr)	$\overline{STROBE}$ ↑ to PORT DATA Float Delay (Mode 2)		510		200	CL = 50 pF
28	TdPD(INT)	PORT DATA Match to $\overline{INT}$ ↓ Delay (Mode 3)		1360		540	
29	TdSTB(INT)	$\overline{STROBE}$ ↑ to $\overline{INT}$↓ Delay		1240		490	

NOTES:
[1] TcC = TwCh + TwCl + TrC + TfC.
[2] Increase TdRI(DO) by 10 ns for each 50 pF increase in load up to 200 pF max.
[3] Increase TdIO(DOI) by 10 ns for each 50 pF, increase in loading up to 200 pF max.
[4] For Mode 2: TwSTB > TsPD(STB).
[5] Increase these values by 2 ns for each 10 pF increase in loading up to 100 pF max.
[6] TsCS(RI) may be reduced. However, the time subtracted from TsCS(RI) will be added to TdRI(DO).
[7] 2.5 TcC > (N-2)TdIEI(IEOf) + TdM1(IEO) + TsIEI(IO) ± TTL Buffer Delay, if any.
[8] $\overline{M1}$ must be active for a minimum of two clock cycles to reset the PIO.
* Timings are preliminary and subject to change.

Zilog

AC and DC Characteristics

Preliminary

September 1983

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information in Product Specifications

Storage Temperature . . . . . -65°C to +150°C

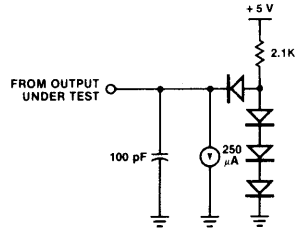
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature range is:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V

*See Ordering Information section in product specifications for package temperature range and product number.



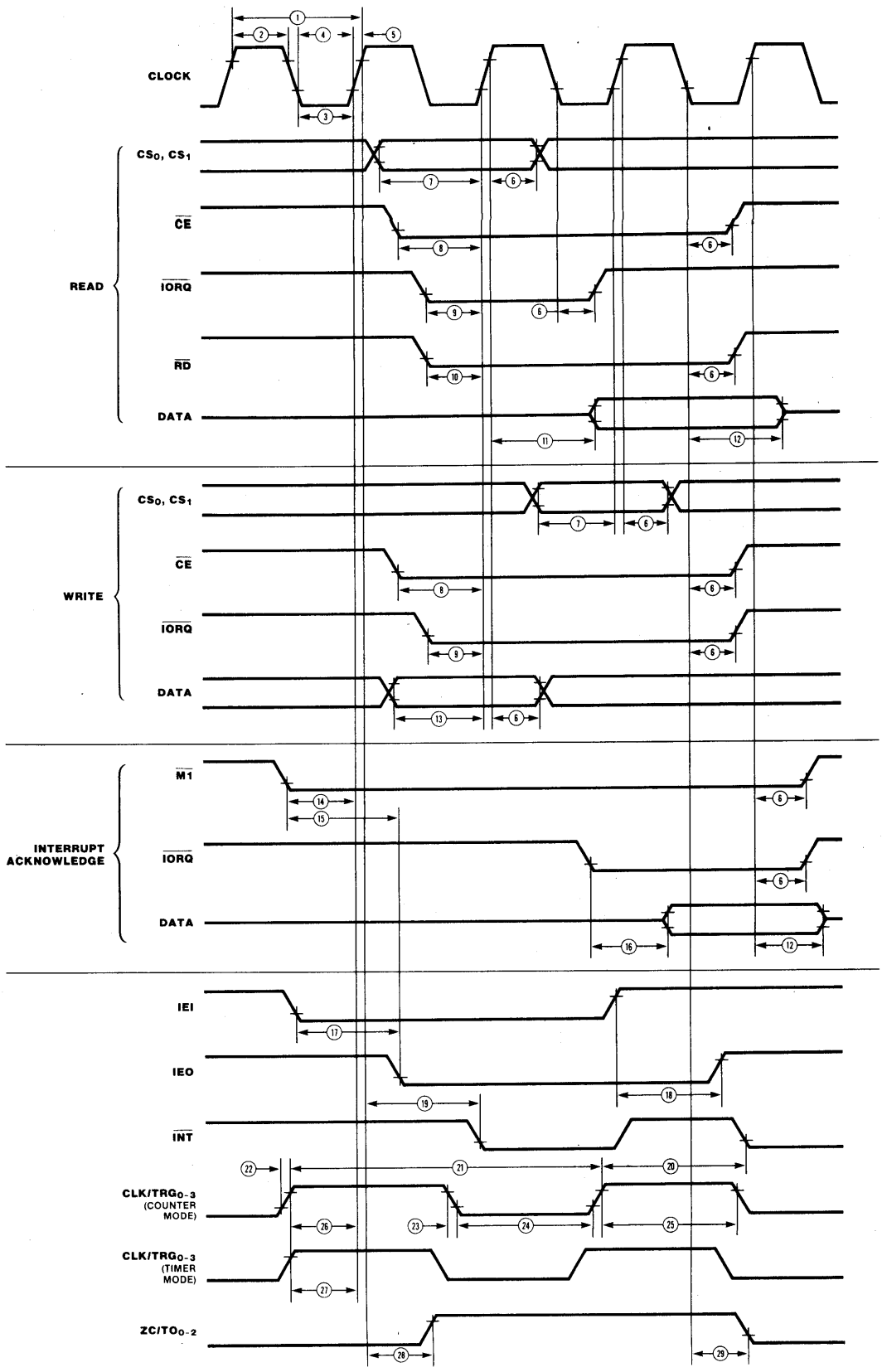
DC Characteristics

Symbol	Parameter	Min	Max	Typical	Unit	Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45		V	
V_{IHC}	Clock Input High Voltage	V_{CC}-0.6	V_{CC}+0.3		V	
V_{IL}	Input Low Voltage	-0.3	+0.8		V	
V_{IH}	Input High Voltage	+2.0	V_{CC}		V	
V_{OL}	Output Low Voltage		+0.4		V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4			V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current		±10		µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10		µA	V_{OUT} = 0.4 to V_{CC}
$I_{L(SY)}$[1]	SYNC Pin Leakage Current		+10/-40		µA	V_{IN} = 0 to V_{CC}
I_{CC}	Power Supply Current:					
	SIO		30	20	mA	
	PIO		20	13	mA	
	CTC		20	13	mA	
I_{OHD}[2]	Darlington Drive Current	-1.5			mA	V_{OH} = 1.5V, R_{EXT} = 390Ω

Over specified temperature and voltage range.

NOTES:
[1] SIO only
[2] CTC and PIO only

Z8330-1 and Z8330-3 Z80L CTC AC Characteristics



Z8330-1 and Z8330-3 Z80L CTC AC Characteristics (Continued)

Number	Symbol	Parameter	Z8330-1 (1.0 MHz) Min	Z8330-1 (1.0 MHz) Max	Z8330-3 (2.5 MHz) Min	Z8330-3 (2.5 MHz) Max	Notes†*
1	TcC	Clock Cycle Time	1000		400		
2	TwCH	Clock Width (High)	470		170		
3	TwCl	Clock Width (Low)	470	2000	170	2000	
4	TfC	Clock Fall Time		30		30	
5	TrC	Clock Rise Time		30		30	
6	Th	All Hold Times	0		0		
7	TsCS(C)	CS to Clock ↑ Setup Time	640		250		
8	TsCE(C)	$\overline{CE}$ to Clock ↑ Setup Time	510		200		
9	TsIO(C)	$\overline{IORQ}$ ↓ to Clock ↑ Setup Time	640		250		
10	TsRD(C)	$\overline{RD}$ ↓ to Clock ↑ Setup Time	610		240		
11	TdC(DO)	Clock ↑ to Data Out Delay		610		240	[2]
12	TdC(DOz)	Clock ↓ to Data Out Float Delay		590		230	
13	TsDI(C)	Data In to Clock ↑ Setup Time	160		60		
14	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	540		210		
15	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (Interrupt immediately preceding $\overline{M1}$)		760		300	[3]
16	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTA Cycle)		860		340	[2]
17	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		490		190	[3]
18	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (After ED Decode)		560		220	[3]
19	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay	((1)+510)		((1)+200)		[4]
20	TdCLK(INT)	CLK/TRG ↑ to $\overline{INT}$↓ tsCTR(C) satisfied	((19)+(26))		((19)+(26))		[5]
		tsCTR(C) not satisfied	((1)+(19)+(26))		((1)+(19)+(26))		[5]
21	TcCTR	CLK/TRG Cycle Time	2TcC		2TcC		[5]
22	TrCTR	CLK/TRG Rise Time		50		50	
23	TfCTR	CLK/TRG Fall Time		50		50	
24	TwCTRl	CLK/TRG Width (Low)	510		200		
25	TwCTRh	CLK/TRG Width (High)	510		200		
26	TsCTR(Cs)	CLK/TRG ↑ to Clock ↑ Setup Time for Immediate Count	760		300		[5]
27	TsCTR(Ct)	CLK/TRG ↑ to Clock ↑ Setup Time for enabling of Prescaler on following clock↑	540		210		[4]
28	TdC(ZC/TOr)	Clock ↑ to ZC/TO ↑ Delay		660		260	
29	TdC(ZC/TOf)	Clock ↓ to ZC/TO ↓ Delay		490		190	

NOTES:

[1] TcC = TwCh + TwCl + TrC + TfC.

[2] Increase delay by 10 ns for each 50 pF increase in loading, 200 pF maximum for data lines, and 100 pF for control lines.

[3] Increase delay by 2 ns for each 10 pF increase in loading, 100 pF maximum.

[4] Timer mode.

[5] Counter mode.

* $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

† Units are nanoseconds unless otherwise specified; parenthetical numbers reference the table number of a parameter, e.g., (1) refers to TcC; timings are preliminary and subject to change.

Zilog

AC and DC Characteristics

Preliminary

September 1983

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information in Product Specifications

Storage Temperature . . . . . -65°C to +150°C

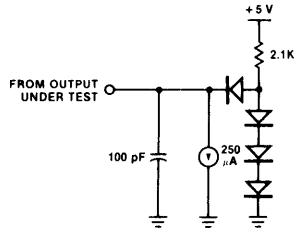
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature range is:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V

*See Ordering Information section in product specifications for package temperature range and product number.



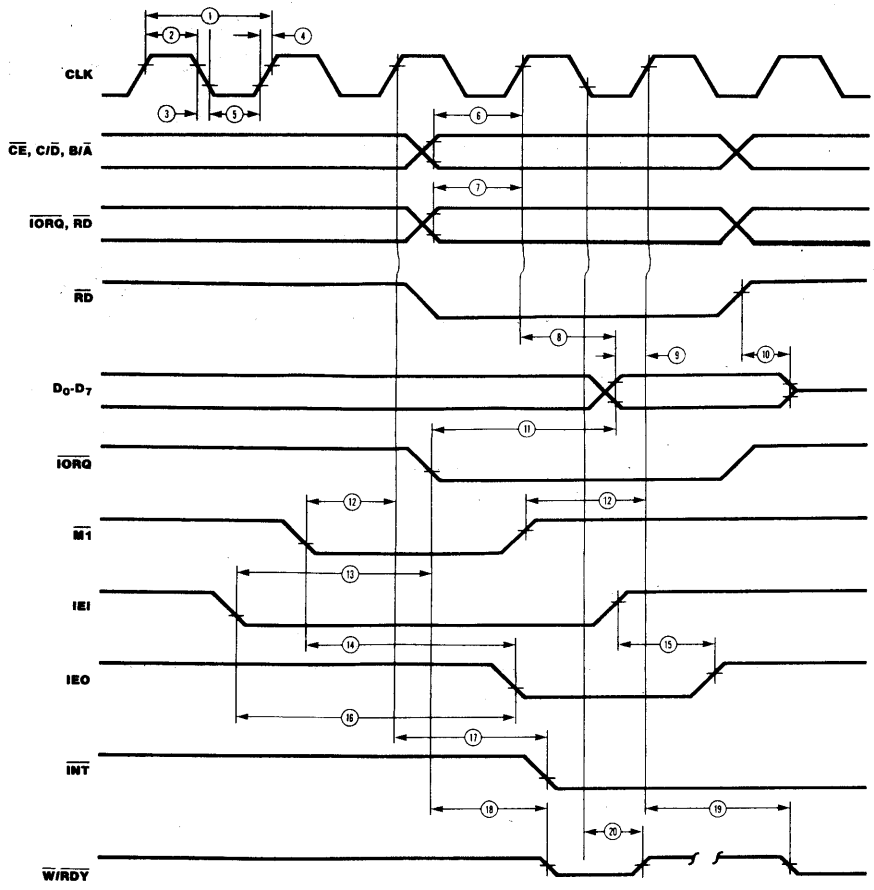
DC Characteristics

Symbol	Parameter	Min	Max	Typical	Unit	Condition
V_{ILC}	Clock Input Low Voltage	-0.3	+0.45		V	
V_{IHC}	Clock Input High Voltage	V_{CC} -0.6	V_{CC} +0.3		V	
V_{IL}	Input Low Voltage	-0.3	+0.8		V	
V_{IH}	Input High Voltage	+2.0	V_{CC}		V	
V_{OL}	Output Low Voltage		+0.4		V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	+2.4			V	I_{OH} = -250 µA
I_{LI}	Input Leakage Current		±10		µA	V_{IN} = 0 to V_{CC}
I_{LO}	3-State Output Leakage Current in Float		±10		µA	V_{OUT} = 0.4 to V_{CC}
$I_{L(SY)}$[1]	SYNC Pin Leakage Current		+10/-40		µA	V_{IN} = 0 to V_{CC}
I_{CC}	Power Supply Current:					
	SIO		30	20	mA	
	PIO		20	13	mA	
	CTC		20	13	mA	
I_{OHD}[2]	Darlington Drive Current	-1.5			mA	V_{OH} = 1.5V, R_{EXT} = 390Ω

Over specified temperature and voltage range.

NOTES:
[1] SIO only
[2] CTC and PIO only

Z8340-1 and Z8340-3 Z80L SIO AC Characteristics

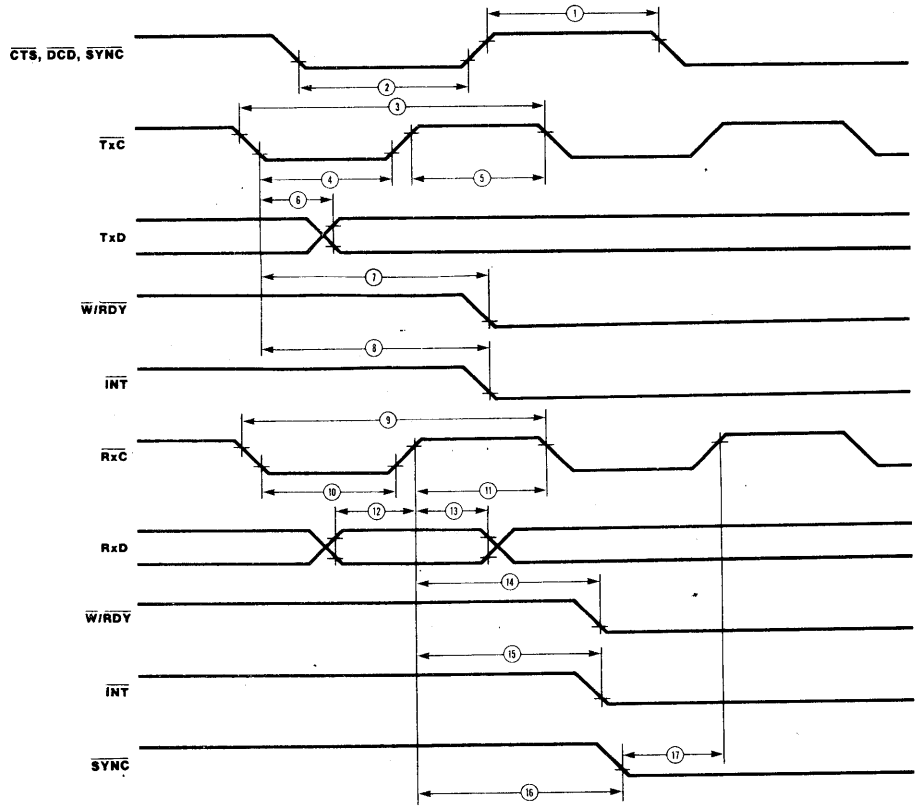


Z8340-1 and Z8340-3 Z80L SIO AC Characteristics (Continued)

Number	Symbol	Parameter	Z8340-1 (1.0 MHz) Min	Z8340-1 (1.0 MHz) Max	Z8340-3 (2.5 MHz) Min	Z8340-3 (2.5 MHz) Max	Notes†
1	TcC	Clock Cycle Time	1000	4000	400	4000	
2	TwCh	Clock Width (High)	470	2000	170	2000	
3	TfC	Clock Fall Time		30		30	
4	TrC	Clock Rise Time		30		30	
5	TwCl	Clock Width (Low)	470	2000	170	2000	
6	TsAD(C)	$\overline{CE}$, C/$\overline{D}$, B/$\overline{A}$ to Clock ↑ Setup Time	410		160		
7	TsCS(C)	$\overline{IORQ}$, $\overline{RD}$ to Clock ↑ Setup Time	610		240		
8	TdC(DO)	Clock ↑ to Data Out Delay		610		240	
9	TsDI(C)	Data In to Clock ↑ Setup (Write or $\overline{M1}$ Cycle)	140		50		
10	TdRD(DOz)	$\overline{RD}$ ↑ to Data Out Float Delay		590		230	
11	TdIO(DOI)	$\overline{IORQ}$ ↓ to Data Out Delay (INTACK Cycle)		860		340	
12	TsM1(C)	$\overline{M1}$ to Clock ↑ Setup Time	540		210		
13	TsIEI(IO)	IEI to $\overline{IORQ}$ ↓ Setup Time (INTACK Cycle)	510		200		
14	TdM1(IEO)	$\overline{M1}$ ↓ to IEO ↓ Delay (interrupt before $\overline{M1}$)		760		300	
15	TdIEI(IEOr)	IEI ↑ to IEO ↑ Delay (after ED decode)		380		150	
16	TdIEI(IEOf)	IEI ↓ to IEO ↓ Delay		380		150	
17	TdC(INT)	Clock ↑ to $\overline{INT}$ ↓ Delay		510		200	
18	TdIO(W/RWf)	$\overline{IORQ}$ ↓ or $\overline{CE}$ ↓ to $\overline{W/RDY}$ ↓ Delay Wait Mode)		760		300	
19	TdC(W/RR)	Clock ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)		310		120	
20	TdC(W/RWz)	Clock ↓ to $\overline{W/RDY}$ Float Delay (Wait Mode)		390		150	
21	Th	Any unspecified Hold when Setup is specified	0		0		

† Units are nanoseconds unless otherwise specified; timings are preliminary and subject to change.

Z8340-1 and Z8340-3 Z80L SIO AC Characteristics (Continued)



Number	Symbol	Parameter	Z8340-1 (1.0 MHz) Min	Z8340-1 (1.0 MHz) Max	Z8340-3 (2.5 MHz) Min	Z8340-3 (2.5 MHz) Max	Notes†
1	TwPh	Pulse Width (High)	500		200		
2	TwPl	Pulse Width (Low)	500		200		
3	TcTxC	$\overline{TxC}$ Cycle Time	1000	∞	400	∞	
4	TwTxCl	$\overline{TxC}$ Width (Low)	460	∞	180	∞	
5	TwTxCh	$\overline{TxC}$ Width (High)	460	∞	180	∞	
6	TdTxC(TxD)	$\overline{TxC}$ ↓ to TxD Delay (x1 Mode)		1000		400	
7	TdTxC(W/RRf)	$\overline{TxC}$ ↓ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	5	9	5	9	Clk Periods*
8	TdTxC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ ↓ Delay	5	9	5	9	Clk Periods*
9	TcRxC	$\overline{RxC}$ Cycle Time	1000	∞	400	∞	
10	TwRxCl	$\overline{RxC}$ Width (Low)	460	∞	180	∞	
11	TwRxCh	$\overline{RxC}$ Width (High)	460	∞	180	∞	
12	TsRxD(RxC)	RxD to $\overline{RxC}$ ↑ Setup Time (x1 Mode)	0		0		
13	ThRxD(RxC)	$\overline{RxC}$ ↑ to RxD Hold Time (x1 Mode)	360		140		
14	TdRxC(W/RRf)	$\overline{RxC}$ ↑ to $\overline{W/RDY}$ ↓ Delay (Ready Mode)	10	13	10	13	Clk Periods*
15	TdRxC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ ↓ Delay	10	13	10	13	Clk Periods*
16	TdRxC(SYNC)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ ↓ Delay (Output Modes)	4	7	4	7	Clk Periods*
17	TsSYNC(RxC)	$\overline{SYNC}$ ↓ to $\overline{RxC}$ ↑ Setup (External Sync Modes)	100		100		

In all modes, the System Clock rate must be at least five times the maximum data rate.
$\overline{RESET}$ must be active a minimum of one complete Clock Cycle.

* System Clock
† Units are nanoseconds unless otherwise specified; timings are preliminary and subject to change.

Z8000

Family

Zilog

Pioneering the Microworld

Zilog Z8000™ Family

A High-Performance 16-Bit Architecture With 32-Bit Migration in Mind

September 1983

A Complete Solution. Continuing the family concept so successfully introduced by its 8-bit Z80 CPU, Zilog devised the Z8000 Family of 16-bit parts. As you would expect from Zilog, this family provides much more than an extension of 8-bit architecture: the Z8000 Family lets you design advanced concepts from the mainframe and minicomputer worlds into microcomputer systems.

And because the Z8000 Family was built around a unifying set of protocols and interconnections, present and future family members are entirely compatible. Your system can grow as your applications mature or expand. A whole range of functions have been planned for from the beginning; the growing family now includes parts to provide memory management, DMA transfer, and extended processing.

System Flexibility. Even the smallest Z8000 systems offer high throughput and easy programming far superior to any existing microprocessor alternative. In mid-range applications, Z8000 components offer very powerful solutions to the design problems of word processing, intelligent terminals, data communications, instrumentation, and process control. In a complex network of multiple processors, smart peripheral components, and a distributed memory configuration, the Z8000 Family provides performance and versatility exceeding that of much larger—and far more expensive—minicomputers.

Higher Throughput, Reduced Cost. The powerful instruction set, high execution speed, regular architecture, and numerous special features of the Z8000 microprocessors dramatically increase system throughput. Intelligent Z8000 peripheral controllers and extended processing units unburden the CPU and boost throughput even further.

Simply put, the Z8000 Family offers more for less money. The Z8000 microprocessors give mid-range minicomputer performance at microprocessor cost. At component prices, Z8000 peripheral controllers perform complex system functions that previously required an entire PC board.

The Z8000 Family is designed for multiple-processor operation—an economical way of greatly increasing system performance. Many special features for multiple Z8000 CPUs facilitate the design of multiple-processor systems that share access to a common memory. The Memory Management Units can dynamically relocate code and protect memory areas. The Z8090/4 Z-UPC Universal Peripheral Controller, a complete slave microcomputer, and the Z8070 Floating Point Emulation Package for high-speed arithmetic, can manipulate data off-line. Asynchronous parts of multiple-processor systems can be joined by the Z8038 Z-FIO FIFO Interface Unit.

An Unmatched CPU. The Z8000 microprocessor is not just a wider data path, more registers, more data types, more addressing modes, more instructions and more addressing space. It brings big-machine concepts to the level of components. Its general-register architecture avoids bottlenecks associated with dedicated or implied registers. Special features support parallel processors, operating systems, compilers, and the implementation of virtual memory.

The Z8000 CPU is also a very fast machine. Its throughput is greater than that of any other 16-bit microprocessor with comparable clock speeds. And the Z8000 CPU is available with speeds ranging from a moderate 4 MHz clock rate that allows you the choice of slow-access, low-cost memories to a high-speed 10 MHz clock rate for high-performance systems. From the four versions of the Z8000 microprocessors, you can select the one best suited to your needs: the Z8001 for large memory applications, the Z8002 for small memory applications, the Z8003 for virtual memory, or the Z8004 for multiprocessors sharing a common, small memory.

How to Manage Your Memory Better. Trends are increasingly toward systems with multiple users, complex programs, security requirements and memories that don't stop growing. These design problems pose questions not sufficiently answered by other microprocessor families.

Exemplifying the Z-Family commitment to advanced architecture, the Z8010 Memory Management Unit (Z-MMU) and the Z8015 Paged Virtual Memory Management Unit (Z-PMMU) both provide flexibility in code segmenter page relocation and sophistication in memory protection rarely found in the microprocessor world. These devices encourage modular software development—a critical factor as programs reach new levels of complexity.

You are free from specifying where information is actually located in physical memory because the Z-MMU and Z-PMMU make software addresses totally independent from the actual physical memory address. While some microprocessor CPUs do have internal CPU relocation registers, they are dedicated and support few segments. These CPUs also restrict memory protection. Not true for the Z-MMU or Z-PMMU. Various configurations of these devices can randomly relocate all 128 segments output by the Z8000 CPU in any of its available memory systems.

For even more sophisticated memory management, the Z8000 microprocessors include a new member that supports virtual memory via an instruction abort mechanism. The Z8003 Virtual Memory Processing Unit (Z-VMPU) can implement either segmented virtual memory that allows demand swapping of segments, or a paged virtual memory in which the unit of memory allocation is a page within a segment.

But the memory management units are more than relocation devices. They offer you a host of memory protection features that allow the system to protect its software from unwanted uses and users. Segments or pages can be specified as read-only to protect them from being overwritten, as system-only to protect the operating system from inadvertent user access, as execute-only, and so on. A write warning zone is especially useful in stack operations so the operating system can deal with growing stacks.

Peripheral Problem Solvers. Z8000 peripheral components are not dumb I/O circuits. They perform intelligent, complicated tasks on their own. They unburden the CPU, reduce bus traffic, and increase system throughput. Complex system tasks that previously required burdensome conglomerations of MSI can now be handled off-line by Z-BUS peripherals with little CPU overhead. Multifunction Z-BUS peripherals are extensively programmable, so each can be precisely tailored to its application.

Counting, timing, and parallel I/O problems seem less tiresome with the Z8036 Z-CIO Counter and Parallel I/O device. It has three 16-bit counter/timers, and three I/O ports. It can even double as a programmable interrupt-priority controller. Data communications are neatly handled by the Z8030 Z-SCC Serial Communication Controller and the Z8033 Z-ASCC Asynchronous Serial Communications Controller, dual-channel multi-protocol components that support between them all popular communications formats. Direct memory access is amply supported by the Z8016 Z-DTC DMA Transfer Controller, a fast dual-channel device that enhances the addressing power of the Z8000 CPU in stand-alone or parallel-processor environments. General-purpose control and data-manipulation problems are smoothly solved by the Z8034 Z-UPC Universal Peripheral Controller, a complete off-line microcomputer-on-a-chip with three I/O ports. This processor executes the same friendly, capable instruction set as our Z8 Microcomputer. Bits and pieces of asynchronous parallel-processing systems are interconnected by the Z8038 Z-FIO FIFO Input/Output, a surprisingly flexible device that can interface any major microprocessor and most peripherals to the Z-BUS. Its buffer depth can be expanded without limit using the Z8060 Z-FIFO.

Where high-speed error detection and correction are essential, the Z8065 Burst Error Processor (Z-BEP) offers a choice of four selectable industry-standard polynomials and three corrections algorithms. It is effective for data rates of up to 20M bits per second. If encryption or decryption of data is necessary, the Z8068 Data Ciphering Processor (Z-DCP) supports three standard ciphering options and key parity check. It can also input, output, and encipher simultaneously. To perform high-speed arithmetic now, the Z8070 Floating Point Emulation Package (with IEEE P754 Standard format) can be implemented with any Z8000 microprocessor. Later this software package will be replaceable by the Z8070 Floating Point Arithmetic Processing Unit itself.

Z8001/2
Z8000™ CPU Central Processing Unit

Zilog

Product Specification

September 1983

Features

- Regular, easy-to-use architecture
- Instruction set more powerful than many minicomputers
- Directly addresses 8M bytes
- Eight user-selectable addressing modes
- Seven data types that range from bits to 32-bit long words and byte and word strings
- System and Normal operating modes
- Separate code, data and stack spaces
- Sophisticated interrupt structure
- Resource-sharing capabilities for multiprocessing systems
- Multi-programming support
- Compiler support
- Memory management and protection provided by Z8010 Memory Management Unit
- 32-bit operations, including signed multiply and divide
- Z-BUS compatible
- 4, 6 and 10 MHz clock rate

General Description

The Z8000 is an advanced high-end 16-bit microprocessor that spans a wide variety of applications ranging from simple stand-alone computers to complex parallel-processing systems. Essentially a monolithic minicomputer central processing unit, the Z8000 CPU is characterized by an instruction set more powerful than many minicomputers; abundant resources in registers, data types, addressing modes and addressing range; and a regular architecture that enhances throughouput by avoiding critical bottlenecks such as implied or dedicated registers.

CPU resources include sixteen 16-bit general-purpose registers, seven data types that range from bits to 32-bit long words and byte and word strings, and eight user-selectable addressing modes. The 110 distinct instruction types can be combined with the various data types and addressing modes to form a powerful set of 414 instructions. Moreover, the instruction set is regular; most instructions can use any of the five main addressing modes and can operate on byte, word and long-word data types.

The CPU can operate in either the system or normal modes. The distinction between these two modes permits privileged operations, thereby improving operating system organization and implementation. Multiprogramming is supported by the "atomic" Test and Set instruction; multiprocessing by a combination of instruction and hardware features; and compilers by multiple stacks, special instructions and addressing modes.

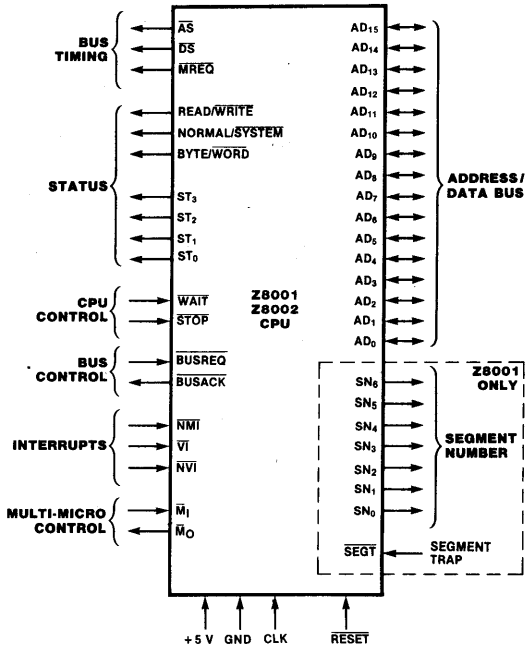


Figure 1. Z8000 CPU Pin Functions

General Description (Continued)

The Z8000 CPU is offered in two versions: the Z8001 48-pin segmented CPU and the Z8002 40-pin non-segmented CPU. The main difference between the two is in addressing range. The Z8001 can directly address 8 megabytes of memory; the Z8002 directly addresses 64 kilobytes. The two operating modes - system and normal - and the distinction between code, data and stack spaces within each mode allows memory extension up to 48 megabytes for the Z8001 and 384 kilobytes for the Z8002.

To meet the requirements of complex, memory-intensive applications, a companion memory-management device is offered for the Z8001. The Z8010 Memory Management Unit manages the large address space by providing features such as segment relocation and memory protection. The Z8001 can be used with or without the Z8010. If used by itself, the Z8001 still provides an 8 megabyte direct addressing range, extendable to 48 megabytes.

The Z8001, Z8002 and Z8010 are fabricated with high-density, high-performance scaled *n-channel silicon-gate depletion-load technology*, and are housed in dual in-line packages.

Register Organization

The Z8000 CPU is a register-oriented machine that offers sixteen 16-bit general-purpose registers and a set of special system registers. All general-purpose registers can be used as accumulators and all but one as index registers or memory pointers.

Register flexibility is created by grouping and overlapping multiple registers (Figures 2 and 3). For byte operations, the first eight 16-bit registers (R0...R7) are treated as sixteen 8-bit registers (RL0, RH0, ... , RL7, RH7). The sixteen 16-bit registers are grouped in pairs (RR0 ... RR14) to form 32-bit long-word registers. Similarly, the register set is grouped in quadruples (RQ0 ... RQ12) to form 64-bit registers.

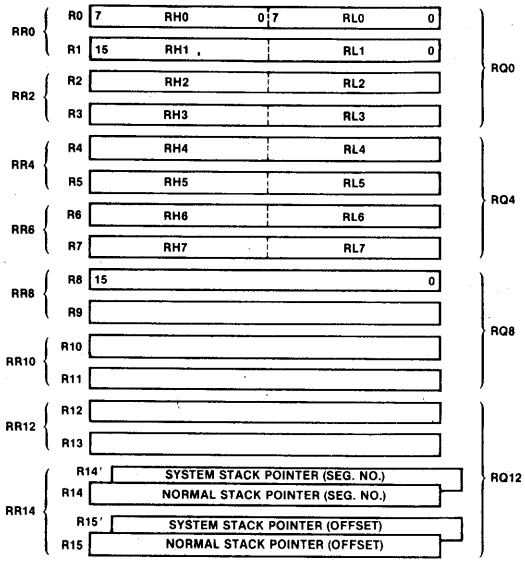


Figure 2. Z8001 General-Purpose Registers

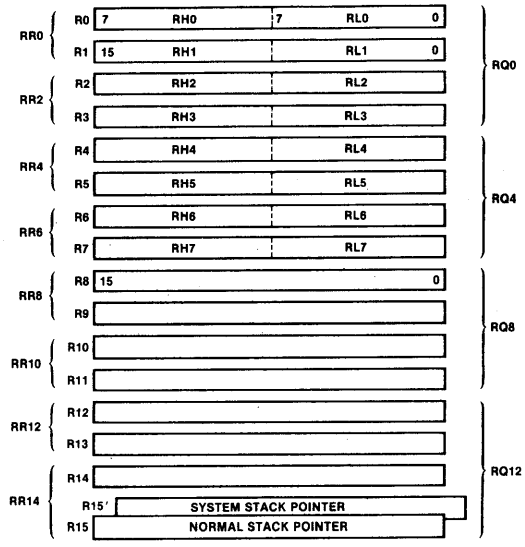


Figure 3. Z8002 General-Purpose Registers

Stacks

The Z8001 and Z8002 can use stacks located anywhere in memory. Call and Return instructions as well as interrupts and traps use implied stacks. The distinction between normal and system stacks separates system information from the application program information. Two stack pointers are available: the system stack pointer and the normal stack pointer. Because they are part of the general-purpose register group, the user can manipulate the stack pointers with any instruction available for register operations.

In the Z8001, register pair RR14 is the implied stack pointer. Register R14 contains the 7-bit segment number and R15 contains the 16-bit offset. In the Z8002, register R15 is the implied 16-bit stack pointer.

Refresh

The Z8000 CPU contains a counter that can be used to automatically refresh dynamic memory. The refresh counter register consists of a 9-bit row counter, a 6-bit rate counter and an enable bit (Figure 4). The 9-bit row counter can address up to 256 rows and is incremented by two each time the rate counter reaches end-of-count. The rate counter determines the time between successive refreshes. It consists of a programmable 6-bit modulo-n prescaler (n = 1 to 64), driven at one-fourth the CPU clock rate. The refresh period can be programmed by 1 to 64 μs with a 4 MHz clock. Refresh can be disabled by programming the refresh enable/disable bit.

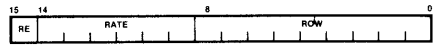


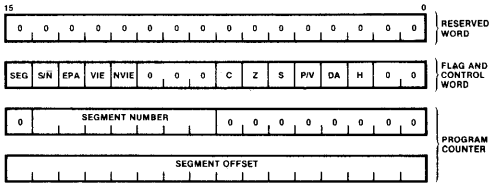
Figure 4. Refresh Counter

Program Status Information

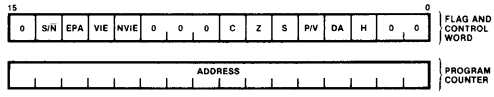
This group of status registers contains the program counter, flags and control bits. When an interrupt or trap occurs, the entire group is saved and a new program status group is loaded.

Figure 5 illustrates how the program status groups of the Z8001 and Z8002 differ. In the non-segmented Z8002, the program status group consists of two words: the program counter (PC), and the flag and control word (FCW). In the segmented Z8001, the program status group consists of four words: a two-word program counter, the flag and control word, and an unused word reserved for future use. Seven bits of the first PC word designate one of the 128 memory segments. The second word supplies the 16-bit offset that designates a memory location within the segment.

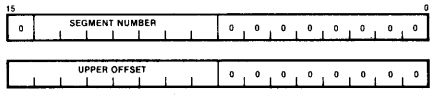
With the exception of the segment enable bit in the Z8001 program status group, the flags and control bits are the same for both CPUs.



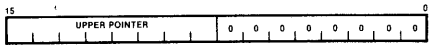
Z8001 Program Status Registers



Z8002 Program Status Registers



Z8001 Program Status Area Pointer



Z8002 Program Status Area Pointer

Figure 5. Z8000 CPU Special Registers

Interrupt and Trap Structure

The Z8000 provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions. Both are processes in a similar manner by the CPU.

The CPU supports three types of interrupts (non-maskable, vectored and non-vectored) and four traps (system call, Extended Process Architecture instruction, privileged instructions and segmentation trap). The vectored and non-vectored interrupts are maskable. Of the four traps, the only external one is the segmentation trap, which is generated by the Z8010.

The remaining traps occur when instructions limited to the system mode are used in the normal mode, or as a result of the System Call instruction, or for an EPA instruction. The descending order of priority for traps and interrupts is: internal traps, non-maskable interrupt, segmentation trap, vectored interrupt and non-vectored interrupt.

When an interrupt or trap occurs, the current program status is automatically pushed on the system stack. The program status consists of the processor status (PC and FCW) plus a 16-bit identifier. The identifier contains the reason or source of the trap or interrupt. For internal traps, the identifier is the first word of the trapped instruction. For external traps or interrupts, the identifier is the vector on the data bus read by the CPU during the interrupt-acknowledge or trap-acknowledge cycle.

After saving the current program status, the new program status is automatically loaded from the program status area in system memory. This area is designated by the program status area pointer (PSAP).

Data Types

Z8000 instructions can operate on bits, BCD digits (4 bits), bytes (8 bits), words (16 bits), long words (32 bits) and byte strings and word strings (up to 64 kilobytes long), and word strings (up to 64 kilobytes long). Bits can be set, reset and tested; digits are used in BCD arithmetic operations; bytes are used for characters or small integer values; words are used for integer values, instructions and non-segmented addresses; long words are used for long integer values and segmented addresses. All data elements except strings can reside either in registers or memory. Strings are stored in memory only.

The basic data element is the byte. The number of bytes used when manipulating a data element is either implied by the operation or - for strings and multiple register operations - explicitly specified in the instruction.

Segmentation and Memory Management

High-level languages, sophisticated operating systems, large programs and data bases, and decreasing memory prices are all accelerating the trend toward larger memory requirements in microcomputer systems. The Z8001 meets this requirement with an eight megabyte addressing space. This large address space is directly accessed by the CPU using a segmented addressing scheme and can be managed by the Z8010 Memory Management Unit.

Segmented Addressing

A segmented addressing space - compared with linear addressing - is closer to the way a programmer uses memory because each procedure and data space resides in its own segment. The 8 megabytes of Z8001 addressing space is divided into 128 relocatable segments up to 64 kilobytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any location relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately. The segmented Z8001 can run any code written for the non-segmented Z8002 in any one of its 128 segments, provided it is set to the non-segmented mode.

In hardware, segmented addresses are contained in a register pair or long-word memory location. The segment number and offset can be manipulated separately or together by all the available word and long-word operations.

When contained in an instruction, a segmented address has two different representations: long offset and short offset. The long offset occupies two words, whereas the short offset requires only one and combines in one word the 7-bit segment number with an 8-bit offset (range 0-256). The short offset mode allows very dense encoding of addresses and minimizes the need for long addresses required by direct accessing of this large address space.

Memory Management

The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical* addresses. The Memory Management Unit takes the logical addresses and transforms them into the *physical* addresses required for accessing the memory (Figure 6). This address transformation process is called relocation. Segment relocation makes user software addresses independent of the physical memory so the user is freed from specifying where information is actually located in the physical memory.

The relocation process is transparent to user software. A translation table in the Memory Management Unit associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit offset is added to the physical base address to obtain the actual physical address. The system may dynamically reload translation tables as tasks are created, suspended or changed.

In addition to supporting dynamic segment relocation, the Memory Management Unit also provides segment protection and other segment management features. The protection features prevent illegal uses of segments, such as writing into a write-protected zone.

Each Memory Management Unit stores 64 segment entries that consist of the segment base address, its attributes, size and status. Segments are variable in size from 256 bytes to 64 kilobytes in increments of 256 bytes. Pairs of Management Units support the 128 segment numbers available for each of the six CPU address spaces. Within an address space, several Management Units can be used to create multiple translation tables.

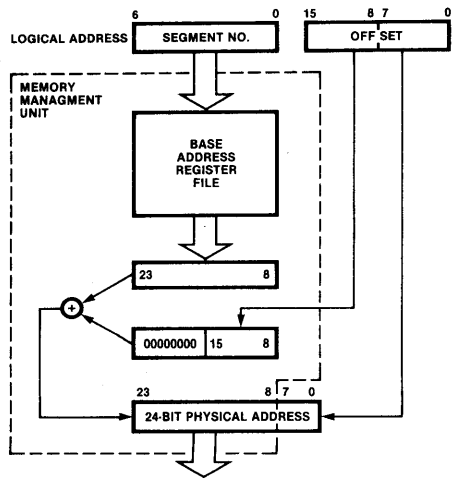


Figure 6. Logical-to-Physical Address Transformation

Extended Processing Architecture

The Zilog Extended Processing Architecture (EPA) provides an extremely flexible and modular approach to expanding both the hardware and software capabilities of the Z8000 CPU. Features of the EPA include:

- Specialized instructions for external processors or software traps may be added to CPU instruction set.
- Increases throughput of the system by using up to four specialized external processors in parallel with the CPU.
- Permits modular design of Z8000-based systems.
- Provides easy management of multiple microprocessor configurations via "single instruction stream" communication.
- Simple interconnection between extended processing units and Z8000 CPU requires no additional external supporting logic.
- Supports debugging of suspect hardware against proven software.
- Standard feature on all Zilog Z8000 CPUs.

Specific benefits include:

- EPUs can be added as the system grows and as EPUs with specialized functions are developed.
- Control of EPUs is accomplished via a "single instruction stream" in the Z8000 CPU, eliminating many significant system software and bus contention management obstacles that occur in other multiprocessor (e.g., master-slave) organization schemes.

The processing power of the Zilog Z8000 16-bit microprocessor can be boosted beyond its intrinsic capability by Extended Processing Architecture. Simply stated, EPA allows the Z8000 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream.

The use of extended processors to boost the main CPU's performance capability has been proven with large mainframe computers and minicomputers. In these systems, specialized functions such as array processing, special input/output processing, and data communications processing are typically assigned to extended processor hardware. These extended processors are complex computers in their own right.

The Zilog Extended Processing Architecture combines the best concepts of these proven performance boosters with the latest in high-density MOS integrated-circuit design. The result is an elegant expansion of design capability—a powerful microprocessor architecture capable of connecting single-chip EPUs that permits very effective parallel processing and makes for a smoothly integrated instruction stream from the Z8000 programmer's point of view. A typical addition to the current Z8000 instruction set might be Floating Point Instructions.

The Extended Processing Units connect directly to the Z8000 BUS (Z-BUS) and continuously monitor the CPU instruction stream. When an extended instruction is detected, the appropriate EPU responds, obtaining or placing data or status information on the Z-BUS using the Z8000-generated control signals and performing its function as directed.

The Z8000 CPU is responsible for instructing the EPU and delivering operands and data to it. The EPU recognizes instructions intended for it and executes them, using data supplied

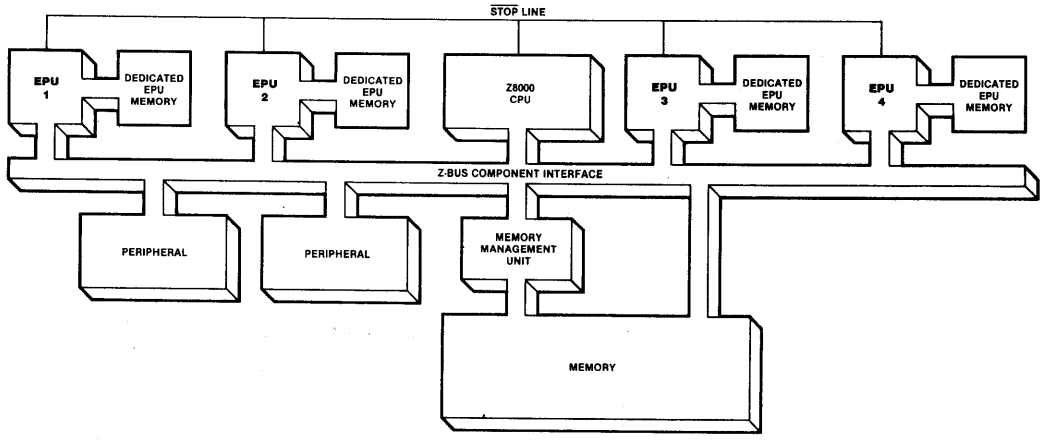


Figure 7. Typical Extended Processor Configuration

Extended Processing Architecture (Continued)

with the instruction and/or data within its internal registers. There are four classes of EPU instructions:

- Data transfers between main memory and EPU registers
- Data transfers between CPU registers and EPU registers
- EPU internal operations
- Status transfers between the EPUs and the Z8000 CPU Flag and Control Word register (FCW)

Four Z8000 addressing modes may be utilized with transfers between EPU registers and the CPU and main memory:

- Register
- Indirect Register
- Direct Address
- Indexed

In addition to the hardware-implemented capabilities of the Extended Processing Architecture, there is an extended instruction trap mechanism to permit software simulation of EPU functions. A control bit in the Z8000 FCW register indicates whether actual EPUs are present or not. If not, when an extended instruction is detected, the Z8000 traps on the instruction, so that a software "trap handler" can emulate the desired EPU function—a very useful development tool. The EPA software trap routine supports the debugging of suspect hardware against proven software. This feature will increase in significance as designers become familiar with the EPA capability of the Z8000 CPU.

This software trap mechanism facilitates the design of systems for later addition of EPUs: initially, the extended function is executed as a trap subroutine; when the EPU is finally attached, the trap subroutine is eliminated and the EPA control bit is set. Application software is unaware of the change.

Extended Processing Architecture also offers protection against extended instruction overlapping. Each EPU connects to the Z8000 CPU via the STOP line so that if an EPU is requested to perform a second extended instruction function before it has completed the previous one, it can put the CPU into the Stop/Refresh state until execution of the previous extended instruction is complete.

EPA and CPU instruction execution are shown in Figure 8. The CPU begins operation by fetching an instruction and determining whether it is a CPU or an EPU command. The EPU meanwhile monitors the Z-BUS for its own instructions. If the CPU encounters an EPU command, it checks to see whether an EPU is present; if not, the EPU may be simulated by an EPU instruction trap software routine; if an EPU is present, the necessary data and/or address is placed on the Z-BUS. If the EPU is free when the instruction and data for it appear, the extended instruction is executed. If the EPU is still processing a previous instruction, it activates the CPU's $\overline{\text{STOP}}$ line to lock the CPU off at the Z-BUS until execution is complete. After the instruction is finished, the EPU deactivates the $\overline{\text{STOP}}$ line and CPU transactions continue.

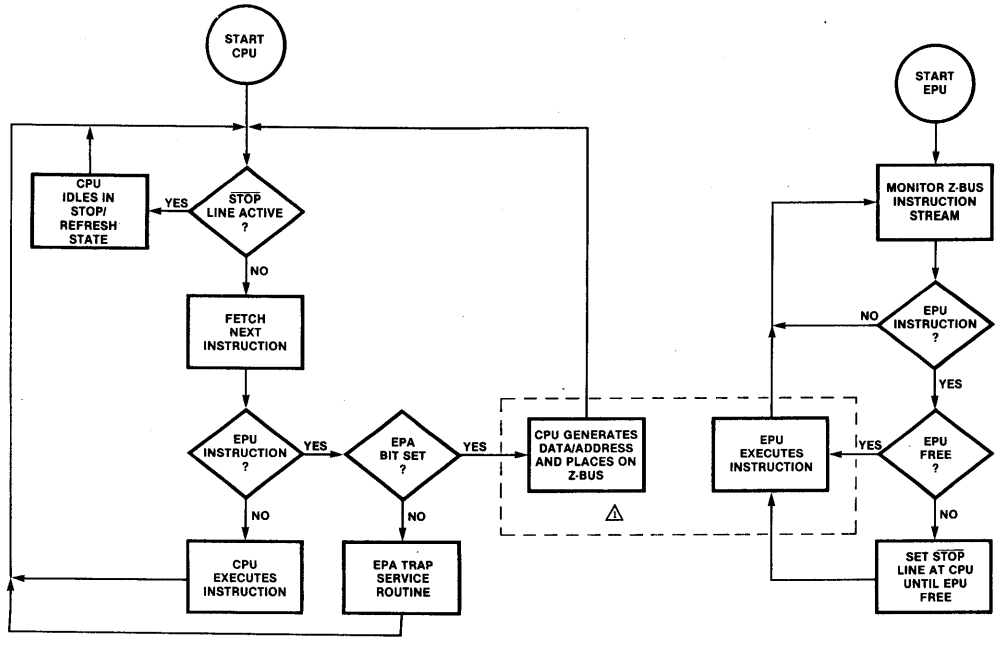


Figure 8. EPA and Z8000 CPU Instruction Execution

Addressing Modes

The information included in Z8000 instructions consists of the function to be performed, the type and size of data elements to be manipulated and the location of the data elements. Locations are designated by register addresses, memory addresses or I/O addresses. The addressing mode of a given instruction defines the address space it references and the method used to compute the address itself. Addressing modes are explicitly specified or implied by the instruction.

Figure 4 illustrates the eight addressing modes: Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Indexed (X), Relative Address (RA), Base Address (BA) and Base Indexed (BX). In general, an addressing mode explicitly specifies either register address space or memory address space. Program memory address space and I/O address space are usually implied by the instruction.

Mode	Operand Addressing			Operand Value
	In the Instruction	In a Register	In Memory	
Register		REGISTER ADDRESS → OPERAND		The content of the register
Immediate	OPERAND			In the instruction
Indirect Register		REGISTER ADDRESS → ADDRESS	→ OPERAND	The content of the location whose address is in the register
Direct Address	ADDRESS		→ OPERAND	The content of the location whose address is in the instruction
Index		REGISTER ADDRESS, BASE ADDRESS; DISPLACEMENT; (+)	→ OPERAND	The content of the location whose address is the address in the instruction, offset by the content of the register
Relative Address		DISPLACEMENT; PC VALUE; (±)	→ OPERAND	The content of the location whose address is the content of the program counter, offset by the displacement in the instruction
Base Address		REGISTER ADDRESS, DISPLACEMENT; BASE ADDRESS; (+)	→ OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the instruction
Base Index		REGISTER ADDRESS, REGISTER ADDRESS; BASE ADDRESS, DISPLACEMENT; (+)	→ OPERAND	The content of the location whose address is the address in the register, offset by the displacement in the register

Figure 9. Addressing Modes

Input/ Output

A set of I/O instructions performs 8-bit or 16-bit transfers betwen the CPU and I/O devices. I/O devices are addressed with a 16-bit I/O port address. The I/O port address is similar to a memory address; however, I/O address space need not be part of the memory address space. I/O port and memory addresses coexist on the same bus lines and they are distinguished by the status outputs.

Two types of I/O instructions are available: standard and special. Each has its own address space. The I/O instructions include a comprehensive set of In, Out and Block I/O instructions for both bytes and words. Special I/O instructions are used for loading and unloading the Memory Management Unit. The status information distinguishes between standard and special I/O references.

Multi-Micro-Processor Support

Multi-microprocessor systems are supported in hardware and software. A pair of CPU pins is used in conjunction with certain instructions to coordinate multiple microprocessors. The Multi-Micro Out pin issues a request for the resource, while the Multi-Micro In pin is used to recognize the state of the resource. Thus, any CPU in a multiple microprocessor system can exclude all other asynchronous CPUs from a critical shared resource.

Multi-microprocessor systems are supported in software by the instructions Multi-Micro Request, Test Multi-Micro In, Set Multi-Micro Out and Reset Multi-Micro Out. In addition, the eight megabyte CPU address space is beneficial in multiple microprocessor systems that have large memory requirements.

Instruction Set Summary

The Z8000 provides the following types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

Load and Exchange

Mnemonics	Operands	Addr. Modes	Clock Cycles* Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CLR	dst	R	7	-	-				**Clear**
CLRB		IR	8	-	-				dst ← 0
		DA	11	12	14				
		X	12	12	15				
EX	R, src	R	6	-	-				**Exchange**
EXB		IR	12	-	-				R ↔ src
		DA	15	16	18				
		X	16	16	19				
LD	R, src	R	3	-	-	5	-	-	**Load into Register**
LDB		IM	7	-	-	11	-	-	R ← src
LDL		IM	5 (byte only)						
		IR	7	-	-	11	-	-	
		DA	9	10	12	12	13	15	
		X	10	10	13	13	13	16	
		BA	14	-	-	17	-	-	
		BX	14	-	-	17	-	-	
LD	dst, R	IR	8	-	-	11	-	-	**Load into Memory** (Store)
LDB		DA	11	12	14	14	15	17	dst ← R
LDL		X	12	12	15	15	15	18	
		BA	14	-	-	17	-	-	
		BX	14	-	-	17	-	-	
LD	dst, IM	IR	11	-	-				**Load Immediate into Memory**
LDB		DA	14	15	17				dst ← IM
		X	15	15	18				

*** NS = Non-Segmented SS = Segmented Short Offset SL = Segmented Long Offset**

	Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Load and Exchange (Continued)	**LDA**	R, src	DA	12	13	15				**Load Address**
			X	13	13	16				R ← source address
			BA	15	–	–				
			BX	15	–	–				
	LDAR	R, src	RA	15	–	–				**Load Address Relative**
										R ← source address
	LDK	R, src	IM	5	–	–				**Load Constant**
										R ← n (n = 0 ... 15)
	LDM	R, src, n	IR	11	–	–	+3n			**Load Multiple**
			DA	14	15	17				R ← src (n consecutive words)
			X	15	15	18				(n = 1 ... 16)
	LDM	dst, R, n	IR	11	–	–	+3n			**Load Multiple** (Store Multiple)
			DA	14	15	17				dst ← R (n consecutive words)
			X	15	15	18				(n = 1 ... 16)
	LDR **LDRB** **LDRL**	R, src	RA	14	–	–	17	–	–	**Load Relative** R ← src (range -32768 ... +32767)
	LDR **LDRB** **LDRL**	dst, R	RA	14	–	–	17	–	–	**Load Relative** (Store Relative) dst ← R (range -32768 ... +32767)
	POP **POPL**	dst, IR	R	8	–	–	12	–	–	**Pop**
			IR	12	–	–	19	–	–	dst ← IR
			DA	16	16	18	23	23	25	Autoincrement contents of R
			X	16	16	19	23	23	26	
	PUSH **PUSHL**	IR, src	R	9	–	–	12	–	–	**Push**
			IM	12	–	–	–	–	–	Autodecrement contents of R
			IR	13	–	–	20	–	–	IR ← src
			DA	14	14	16	21	21	23	
			X	14	14	17	21	21	24	
Arithmetic	**ADC** **ADCB**	R, src	R	5	–	–				**Add with Carry** R ← R + src + carry
	ADD **ADDB** **ADDL**	R, src	R	4	–	–	8	–	–	**Add**
			IM	7	–	–	14	–	–	R ← R + src
			IR	7	–	–	14	–	–	
			DA	9	10	12	15	16	18	
			X	10	10	13	16	16	19	
	CP **CPB** **CPL**	R, src	R	4	–	–	8	–	–	**Compare with Register**
			IM	7	–	–	14	–	–	R - src
			IR	7	–	–	14	–	–	
			DA	9	10	12	15	16	18	
			X	10	10	13	16	16	19	
	CP **CPB**	dst, IM	IR	11	–	–				**Compare with Immediate**
			DA	14	15	17				dst - IM
			X	15	15	18				
	DAB	dst	R	5	–	–				**Decimal Adjust**
	DEC **DECB**	dst, n	R	4	–	–				**Decrement by n**
			IR	11	–	–				dst ← dst - n
			DA	13	14	16				(n = 1 ... 16)
			X	14	14	17				

Arithmetic (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
DIV **DIVL**	R, src	R	107	-	-	744	-	-	**Divide** (signed) Word: $R_{n+1} \leftarrow R_{n,n+1}$ + src $R_n \leftarrow$ remainder Long Word: $R_{n+2,n+3} \leftarrow R_{n\ldots n+3}$ + src $R_{n,n+1} \leftarrow$ remainder
		IM	107	-	-	744	-	-	
		IR	107	107	107	744	744	744	
		DA	108	109	111	745	746	748	
		X	109	109	112	746	746	749	
EXTS **EXTSB** **EXTSL**	dst	R	11	-	-	11	-	-	**Extend Sign** Extend sign of low order half of dst through high order half of dst
INC **INCB**	dst, n	R	4	-	-				**Increment by n** dst ← dst + n (n = 1 ... 16)
		IR	11	-	-				
		DA	13	14	16				
		X	14	14	17				
MULT **MULTL**	R, src	R	70	-	-	282*	-	-	**Multiply** (signed) Word: $R_{n,n+1} \leftarrow R_{n+1}$ • src Long Word: $R_{n\ldots n+3} \leftarrow R_{n+2,\,n+3}$ *Plus seven cycles for each 1 in the multiplicand
		IM	70	-	-	282*	-	-	
		IR	70	-	-	282*	-	-	
		DA	71	72	74	283*	284*	286*	
		X	72	72	75	284*	284*	287*	
NEG **NEGB**	dst	R	7	-	-				**Negate** dst ← 0 - dst
		IR	12	-	-				
		DA	15	16	18				
		X	16	16	19				
SBC **SBCB**	R, src	R	5	-	-				**Subtract with Carry** R ← R - src - carry
SUB **SUBB** **SUBL**	R, src	R	4	-	-	8	-	-	**Subtract** R ← R - src
		IM	7	-	-	14	-	-	
		IR	7	-	-	14	-	-	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

Logical

AND **ANDB**	R, src	R	4	-	-				**AND** R ← R AND src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				
COM **COMB**	dst	R	7	-	-				**Complement** dst ← NOT dst
		IR	12	-	-				
		DA	15	16	18				
		X	16	16	19				
OR **ORB**	R, src	R	4	-	-				**OR** R ← R OR src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				
TCC **TCCB**	cc, dst	R	5	-	-				**Test Condition Code** Set LSB if cc is true
TEST **TESTB** **TESTL**	dst	R	7	-	-	13	-	-	**Test** dst OR 0
		IR	8	-	-	13	-	-	
		DA	11	12	14	16	17	19	
		X	12	12	15	17	17	20	
XOR **XORB**	R, src	R	4	-	-				**Exclusive OR** R ← R XOR src
		IM	7	-	-				
		IR	7	-	-				
		DA	9	10	12				
		X	10	10	13				

	Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Program Control	**CALL**	dst	IR	10	-	15				**Call Subroutine** Autodecrement SP @ SP ← PC PC ← dst
			DA	12	18	20				
			X	13	18	21				
	CALR	dst	RA	10	-	15				**Call Relative** Autodecrement SP @ SP ← PC PC ← PC + dst (range -4094 to +4096)
	DJNZ **DBJNZ**	R, dst	RA	11	-	-				**Decrement and Jump if Non-Zero** R ← R - 1 If R ≠ 0: PC ← PC + dst (range -254 to 0)
	IRET*	-	-	13	-	16				**Interrupt Return** PS ← @ SP Autoincrement SP
	JP	cc, dst	IR	10	-	15	(taken)			**Jump Conditional** If cc is true: PC ← dst
			IR	7	-	7	(not taken)			
			DA	7	8	10				
			X	8	8	11				
	JR	cc, dst	RA	6	-	-				**Jump Conditional Relative** If cc is true: PC ← PC + dst (range -256 to +254)
	RET	cc	-	10	-	13	(taken)			**Return Conditional** If cc is true: PC ← @ SP Autoincrement SP
				7	-	7	(not taken)			
	SC	src	IM	33	-	39				**System Call** Autodecrement SP @ SP ← old PS Push instruction PS ← System Call PS
Bit Manipulation	**BIT** **BITB**	dst, b	R	4	-	-				**Test Bit Static** Z flag ← NOT dst bit specified by b
			IR	8	-	-				
			DA	10	11	13				
			X	11	11	14				
	BIT **BITB**	dst, R	R	10	-	-				**Test Bit Dynamic** Z flag ← NOT dst bit specified by contents of R
	RES **RESB**	dst, b	R	4	-	-				**Reset Bit Static** Reset dst bit specified by b
			IR	11	-	-				
			DA	13	14	16				
			X	14	14	17				
	RES **RESB**	dst, R	R	10	-	-				**Reset Bit Dynamic** Reset dst bit specified by contents R
	SET **SETB**	dst, b	R	4	-	-				**Set Bit Static** Set dst bit specified by b
			IR	11	-	-				
			DA	13	14	16				
			X	14	14	17				
	SET **SETB**	dst, R	R	10	-	-				**Set Bit Dynamic** Set dst bit specified by contents of R
	TSET **TSETB**	dst	R	7	-	-				**Test and Set** S flag ← MSB of dst dst ← all 1s
			IR	11	-	-				
			DA	14	15	17				
			X	15	15	18				

*Privileged instruction. Executed in system mode only.

	Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Rotate and Shift	**RL** **RLB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left** by n bits (n = 1, 2)
	RLC **RLCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Left through Carry** by n bits (n = 1, 2)
	RLDB	R, src	R	9	-	-				**Rotate Digit Left**
	RR **RRB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right** by n bits (n = 1, 2)
	RRC **RRCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right through Carry** by n bits (n = 1, 2)
	RRDB	R, src	R	9	-	-				**Rotate Digit Right**
	SDA **SDAB** **SDAL**	dst, R	R	(15 + 3 n)			(15 + 3 n)			**Shift Dynamic Arithmetic** Shift dst left or right by contents of R
	SDL **SDLB** **SDLL**	dst, R	R	(15 + 3 n)			(15 + 3 n)			**Shift Dynamic Logical** Shift dst left or right by contents of R
	SLA **SLAB** **SLAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Arithmetic** by n bits
	SLL **SLLB** **SLLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Left Logical** by n bits
	SRA **SRAB** **SRAL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Arithmetic** by n bits
	SRL **SRLB** **SRLL**	dst, n	R	(13 + 3 n)			(13 + 3 n)			**Shift Right Logical** by n bits
Block Transfer and String Manipulation	**CPD** **CPDB**	R_X, src, R_Y, cc	IR	20	-	-				**Compare and Decrement** R_X - src Autodecrement src address $R_Y \leftarrow R_Y - 1$
	CPDR **CPDRB**	R_X, src, R_Y, cc	IR	(11 + 9 n)						**Compare, Decrement and Repeat** R_X - src Autodecrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
	CPI **CPIB**	R_X, src, R_Y, cc	IR	20	-	-				**Compare and Increment** R_X - src Autoincrement src address $R_Y \leftarrow R_Y - 1$
	CPIR **CPIRB**	R_X, src, R_Y, cc	IR	(11 + 9 n)						**Compare, Increment and Repeat** R_X - src Autoincrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
	CPSD **CPSDB**	dst, src, R, cc	IR	25	-	-				**Compare String and Decrement** dst - src Autodecrement dst and src addresses R ← R - 1

Block Transfer and String Manipulation (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CPSDR **CPSDRB**	dst, src, R, cc	IR	(11 + 14 n)						**Compare String, Decr. and Repeat** dst - src Autodecrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
CPSI **CPSIB**	dst, src, R, cc	IR	25	-	-				**Compare String and Increment** dst - src Autoincrement dst and src addresses R ← R - 1
CPSIR **CPSIRB**	dst, src, R, cc	IR	(11 + 14 n)						**Compare String, Incr. and Repeat** dst - src Autoincrement dst and src addresses R ← R - 1 Repeat until cc is true or R = 0
LDD **LDDB**	dst, src, R	IR	20	-	-				**Load and Decrement** dst ← src Autodecrement dst and src addresses R ← R - 1
LDDR **LDDRB**	dst, src, R	IR	(11 + 9 n)						**Load, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R - 1 Repeat until R = 0
LDI **LDIB**	dst, src, R	IR	20	-	-				**Load and Increment** dst ← src Autoincrement dst and src addresses R ← R - 1
LDIR **LDIRB**	dst, src, R	IR	(11 + 9 n)						**Load, Increment and Repeat** dst ← src Autoincrement dst and src addresses R ← R - 1 Repeat until R = 0
TRDB	dst, src, R	IR	25	-	-				**Translate and Decrement** dst ← src (dst) Autodecrement dst address R ← R - 1
TRDRB	dst, src, R	IR	(11 + 14 n)						**Translate, Decrement and Repeat** dst ← src (dst) Autodecrement dst address R ← R - 1 Repeat until R = 0
TRIB	dst, src, R	IR	25	-	-				**Translate and Increment** dst ← src (dst) Autoincrement dst address R ← R - 1
TRIRB	dst, src, R	IR	(11 + 14 n)						**Translate, Increment and Repeat** dst ← src (dst) Autoincrement dst address R ← R - 1 Repeat until R = 0
TRTDB	src 1, src 2, R	IR	25	-	-				**Translate and Test, Decrement** RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1

Block Transfer and String Manipulation (Continued)	Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
	TRTDRB	src1, src2, R	IR	(11 + 14 n)						**Translate and Test, Decr. and Repeat** RH1 ← src 2 (src 1) Autodecrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
	TRTIB	src1, src2, R	IR	25						**Translate and Test, Increment** RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1
	TRTIRB	src1, src2, R	IR	(11 + 14 n)						**Translate and Test, Incr. and Repeat** RH1 ← src 2 (src 1) Autoincrement src 1 address R ← R - 1 Repeat until R = 0 or RH1 = 0
Input/ Output	**IN***	R, src	IR	10	-	-				**Input** R ← src
	INB*		DA	12	-	-				
	IND*	dst, src, R	IR	21	-	-				**Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
	INDB*									
	INDR*	dst, src, R	IR	(11 + 10 n)						**Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
	INDRB*									
	INI*	dst, src, R	IR	21	-	-				**Input and Increment** dst ← src Autoincrement dst address R ← R - 1
	INIB*									
	INIR*	dst, src, R	IR	(11 + 10 n)						**Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
	INIRB*									
	OUT*	dst, R	IR	10	-	-				**Output** dst ← R
	OUTB*		DA	12	-	-				
	OUTD*	dst, src, R	IR	21	-	-				**Output and Decrement** dst ← src Autodecrement src address R ← R - 1
	OUTDB*									
	OTDR*	dst, src, R	IR	(11 + 10 n)						**Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
	OTDRB*									
	OUTI*	dst, src, R	IR	21	-	-				**Output and Increment** dst ← src Autoincrement src address R ← R - 1
	OUTIB*									
	OTIR*	dst, src, R	IR	(11 + 10 n)						**Output, Increment and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0
	OTIRB*									

*Privileged instructions. Executed in system mode only.

	Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
Input/Output (Continued)	**SIN*** **SINB***	R, src	DA	12	-	-				**Special Input** R ← src
	SIND* **SINDB***	dst, src, R	IR	21	-	-				**Special Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
	SINDR* **SINDRB***	dst, src, R	IR	(11 + 10 n)						**Special Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
	SINI* **SINIB***	dst, src, R	IR	21	-	-				**Special Input and Increment** dst ← src Autoincrement dst address R ← R - 1
	SINIR* **SINIRB***	dst, src, R	IR	(11 + 10 n)						**Special Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
	SOUT* **SOUTB***	dst, src	DA	12	-	-				**Special Output** dst ← src
	SOUTD* **SOUTDB***	dst, src, R	IR	21	-	-				**Special Output and Decrement** dst ← src Autodecrement src address R ← R - 1
	SOTDR* **SOTDRB***	dst, src, R	IR	(11 + 10 n)						**Special Output, Decr. and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
	SOUTI* **SOUTIB***	dst, src, R	IR	21	-	-				**Special Output and Increment** dst ← src Autoincrement src address R ← R - 1
	SOTIR* **SOTIRB***	dst, src, R	R	(11 + 10 n)						**Special Output, Incr. and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0
CPU Control	**COMFLG**	flags	-	7	-	-				**Complement Flag** (Any combination of C, Z, S, P/V)
	DI*	int	-	7	-	-				**Disable Interrupt** (Any combination of NVI, VI)
	EI*	int	-	7	-	-				**Enable Interrupt** (Any combination of NVI, VI)
	HALT*	-	-	(8 + 3 n)						**HALT**
	LDCTL*	CTLR, src	R	7	-	-				**Load into Control Register** CTLR ← src
	LDCTL*	dst, CTLR	R	7	-	-				**Load from Control Register** dst ← CTLR

*Privileged instructions. Executed in system mode only.

CPU Control (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
LDCTLB	FLGR, src	R	7	-	-				**Load into Flag Byte Register** FLGR ← src
LDCTLB	dst, FLGR	R	7	-	-				**Load from Flag Byte Register** dst ← FLGR
LDPS*	src	IR	12	-	16				**Load Program Status** PS ← src
		DA	16	20	22				
		X	17	20	23				
MBIT*	-	-	7	-	-				**Test Multi-Micro Bit** Set S if $\overline{M_I}$ is Low; reset S if $\overline{M_I}$ is High.
MREQ*	dst	R	(12 + 7 n)						**Multi-Micro Request**
MRES*	-	-	5	-	-				**Multi-Micro Reset**
MSET*	-	-	5	-	-				**Multi-Micro Set**
NOP	-	-	7	-	-				**No Operation**
RESFLG	flag	-	7	-	-				**Reset Flag** (Any combination of C, Z, S, P/V)
SETFLG	flag	-	7	-	-				**Set Flag** (Any combination of C, Z, S, P/V)

*Privileged instructions. Executed in system mode only.

Condition Codes

Code	Meaning	Flag Settings	CC Field
	Always false	-	0000
	Always true	-	1000
Z	Zero	Z = 1	0110
NZ	Not zero	Z = 0	1110
C	Carry	C = 1	0111
NC	No Carry	C = 0	1111
PL	Plus	S = 0	1101
MI	Minus	S = 1	0101
NE	Not equal	Z = 0	1110
EQ	Equal	Z = 1	0110
OV	Overflow	P/V = 1	0100
NOV	No overflow	P/V = 0	1100
PE	Parity is even	P/V = 1	0100
PO	Parity is odd	P/V = 0	1100
GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001
LT	Less than (signed)	(S XOR P/V) = 1	0001
GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010
LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010
UGE	Unsigned greater than or equal	C = 0	1111
ULT	Unsigned less than	C = 1	0111
UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011
ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Note that some condition codes have identical flag settings and binary fields in the instruction:
Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO

Status Code Lines

ST_3-ST_0	Definition	ST_3-ST_0	Definition
0 0 0 0	Internal operation	1 0 0 0	Data memory request
0 0 0 1	Memory refresh	1 0 0 1	Stack memory request
0 0 1 0	I/O reference	1 0 1 0	Data memory request (EPU)
0 0 1 1	Special I/O reference (e.g., to an MMU)	1 0 1 1	Stack memory request (EPU)
0 1 0 0	Segment trap acknowledge	1 1 0 0	Program reference, nth word
0 1 0 1	Non-maskable interrupt acknowledge	1 1 0 1	Instruction fetch, first word
0 1 1 0	Non-vectored interrupt acknowledge	1 1 1 0	Extension processor transfer
0 1 1 1	Vectored interrupt acknowledge	1 1 1 1	Reserved

Pin Description

AD_0-AD_{15}. *Address/Data* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for I/O and to address memory.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates addresses are valid.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). A Low on this line indicates the CPU has relinquished control of the bus.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). This line must be driven Low to request the bus from the CPU.

B/$\overline{W}$. *Byte/Word* (output, Low = Word, 3-state). This signal defines the type of memory reference on the 16-bit address/data bus.

CLK. *System Clock* (input). CLK is a 5V single-phase time-base input.

$\overline{DS}$. *Data Strobe* (output, active Low, 3-state). This line times the data in and out of the CPU.

$\overline{MREQ}$. *Memory Request* (output, active Low, 3-state). A Low on this line indicates that the address/data bus holds a memory address.

$\overline{M}_I$, $\overline{M}_O$. *Multi-Micro In, Multi-Micro Out* (input and output, active Low). These two lines form a resource-request daisy chain that allows one CPU in a multi-microprocessor system to access a shared resource.

$\overline{NMI}$. *Non-Maskable Interrupt* (edge triggered, input, active Low). A high-to-low transition on $\overline{NMI}$ requests a non-maskable interrupt. The NMI interrupt has the highest priority of the three types of interrupts.

N/$\overline{S}$. *Normal/System Mode* (output, Low = System Mode, 3-state). N/$\overline{S}$ indicates the CPU is in the normal or system mode.

$\overline{NVI}$. *Non-Vectored Interrupt* (input, active Low). A Low on this line requests a non-vectored interrupt.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the CPU.

R/$\overline{W}$. *Read/Write* (output, Low = Write, 3-state). R/W indicates that the CPU is reading from or writing to memory or I/O.

$\overline{SEGT}$. *Segment Trap* (input, active Low). The Memory Management Unit interrupts the CPU with a Low on this line when the MMU detects a segmentation trap. Input on Z8001 only.

SN_0-SN_6. *Segment Number* (outputs, active High, 3-state). These lines provide the 7-bit segment number used to address one of 128 segments by the Z8010 Memory Management Unit. Output by the Z8001 only.

ST_0-ST_3. *Status* (outputs, active High, 3-state). These lines specify the CPU status (see table).

$\overline{STOP}$. *Stop* (input, active Low). This input can be used to single-step instruction execution.

$\overline{VI}$. *Vectored Interrupt* (input, active Low). A Low on this line requests a vectored interrupt.

$\overline{WAIT}$. *Wait* (input, active Low). This line indicates to the CPU that the memory or I/O device is not ready for data transfer.

Reserved. Do not connect.

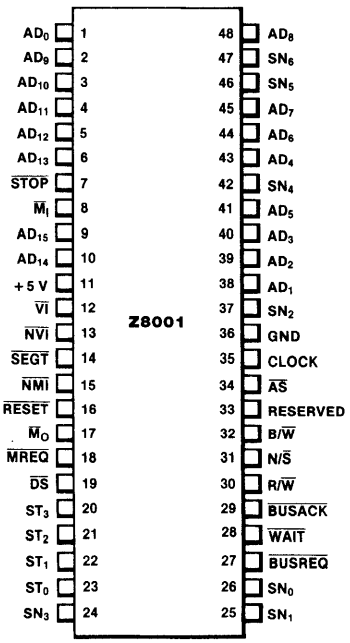


Figure 10. Z8001 Pin Assignments

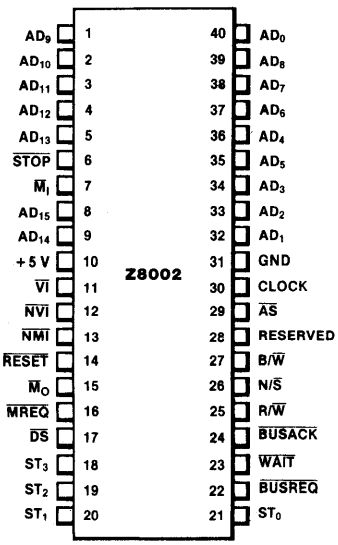


Figure 11. Z8002 Pin Assignments

Z8000 CPU Timing

The Z8000 CPU executes instructions by stepping through sequences of basic machine cycles, such as memory read or write, I/O device read or write, interrupt acknowledge, and internal execution. Each of these basic cycles requires three to ten clock cycles to execute. Instructions that require more clock cycles to execute are broken up into several machine cycles. Thus no machine cycle is longer than ten clock cycles and fast response to a Bus Request is guaranteed.

The instruction opcode is fetched by a normal memory read operation. A memory refresh cycle can be inserted just after the completion of any first instruction fetch (IF_1) cycle and can also be inserted while the following instructions are being executed: MULT, MULTL, DIV, DIVL, HALT, all Shift instructions, all Block Move instructions, and the Multi-Micro Request instruction (MREQ).

The following timing diagrams show the relative timing relationships of all CPU signals during each of the basic operations. When a machine cycle requires additional clock cycles for CPU internal operation, one to five clock cycles are added. Memory and I/O read and write, as well as interrupt acknowledge cycles, can be extended by activating the $\overline{WAIT}$ input. *For exact timing information, refer to the composite timing diagram.*

Note that the $\overline{WAIT}$ input is not synchronized in the Z8000 and that the setup and hold times for $\overline{WAIT}$ relative to the clock must be met. If asynchronous $\overline{WAIT}$ signals are generated, they must be synchronized with the CPU clock before entering the Z8000.

Memory Read and Write

Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 outputs. During a memory read cycle, a 16-bit address is placed on the AD_0-AD_{15} outputs early in the first clock period, as shown in Figure 12. (In the Z8001,

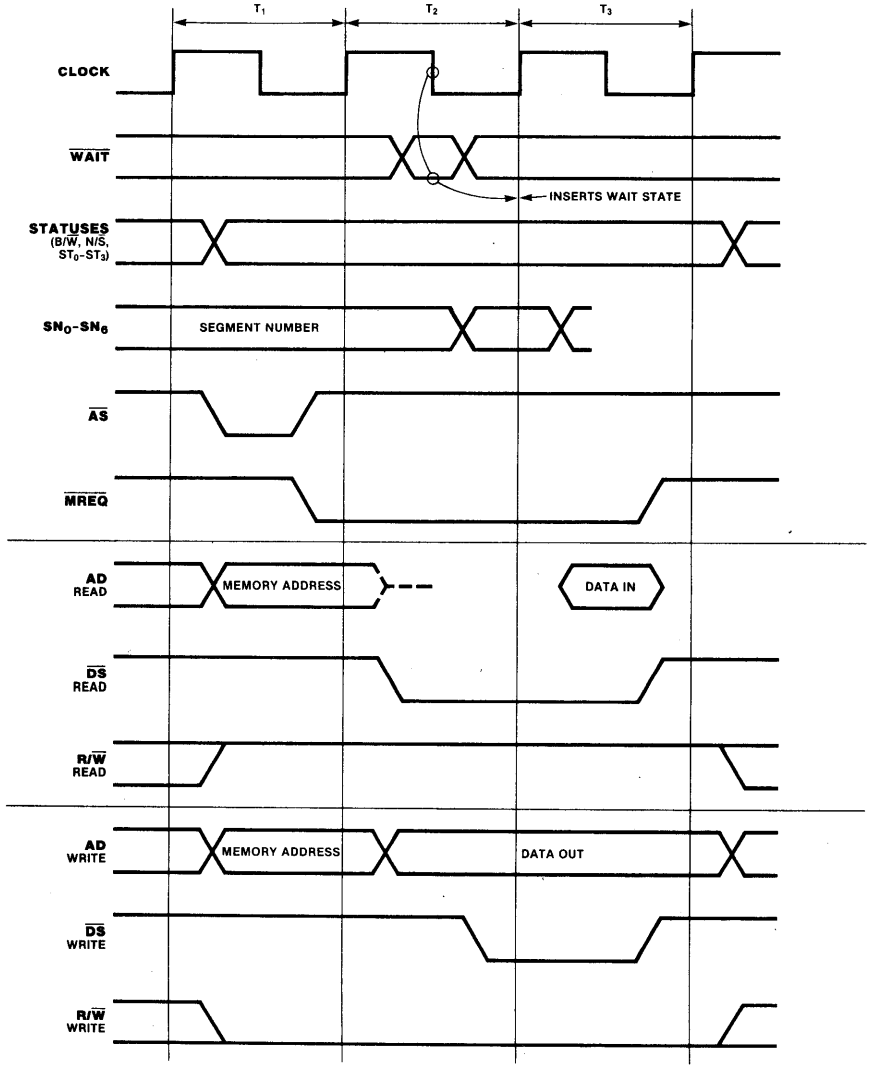


Figure 12. Memory Read and Write Timing

Memory Read and Write (Continued)

the 7-bit segment number is output on SN_0-SN_6 one clock period earlier than the 16-bit address offset to compensate for the delay in the memory management circuitry.)

A valid address is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The state of the $\overline{WAIT}$ input is sampled in the middle of the second clock cycle by the falling edge of Clock. If $\overline{WAIT}$ is Low, an additional clock period is added between T_2 and T_3. $\overline{WAIT}$ is sampled again in the middle of this wait cycle, and additional wait states can be inserted. This allows interfacing slow memories. No control outputs change during wait states.

Although Z8000 memory is word organized, memory is addressed as bytes. All instructions are word-aligned, using even addresses. Within a 16-bit word, the most significant byte (D_8-D_{15}) is addressed by the low-order address (A_0 = Low), and the least significant byte (D_0-D_7) is addressed by the high-order address (A_0 = High).

Input/ Output

I/O timing is similar to memory read/write timing, except that one wait state is automatically inserted between T_2 and T_3 (Figure 13). Both the segmented Z8001 and the non-segmented Z8002 use 16-bit I/O addresses.

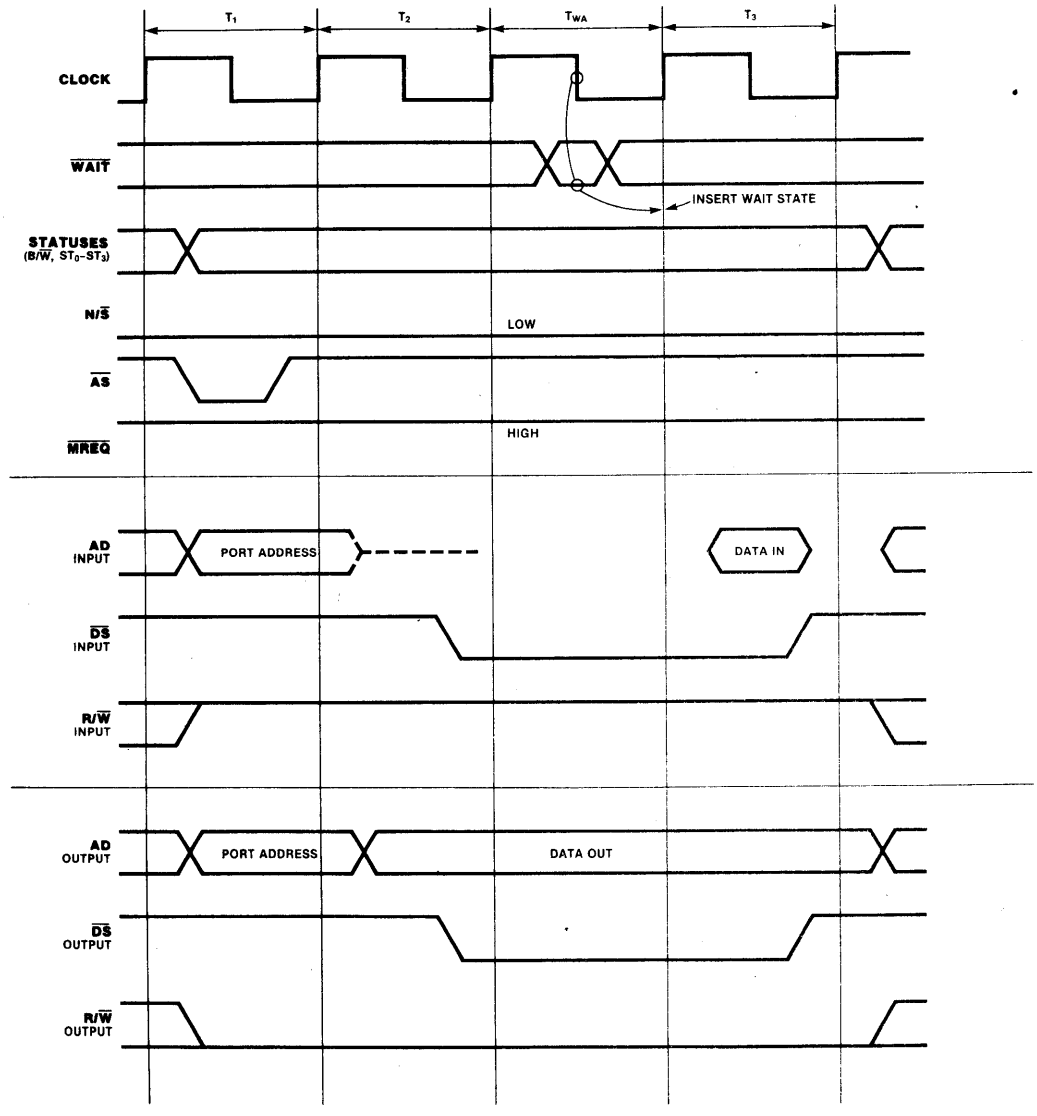


Figure 13. Input/Output Timing

Interrupt and Segment Trap Request and Acknowledge

The Z8000 CPU recognizes three interrupt inputs (non-maskable, vectored and non-vectored) and a segmentation trap input. Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge detected and sets the internal NMI latch. The $\overline{VI}$, $\overline{NVI}$ and $\overline{SEGT}$ inputs as well as the state of the internal NMI latch are sampled at the beginning of T_3 in the last machine cycle of any instruction.

In response to an interrupt or trap, the subsequent IF_1 cycle is exercised, but ignored. The internal state of the CPU is not altered and the instruction will be refetched and executed after the return from the interrupt routine. The program counter is not updated, but and the system stack pointer is decremented in preparation for pushing starting information onto the system stack.

The next machine cycle is the interrupt acknowledge cycle. This cycle has five automatic wait states, with additional wait states possible, as shown in Figure 14.

After the last wait state, the CPU reads the information on AD_0–AD_{15} and stores it temporarily, to be saved on the stack later in the acknowledge sequence. This word identifies the source of the interrupt or trap. For the non-vectored and non-maskable interrupts, all 16 bits can represent peripheral device status information. For the vectored interrupt, the low byte is the jump vector, and the high byte can be extra user status. For the segmentation trap, the *high* byte is the Memory Management Unit identifier and the *low* byte is undefined.

After the acknowledge cycle, the N/S output indicates the automatic change to system mode.

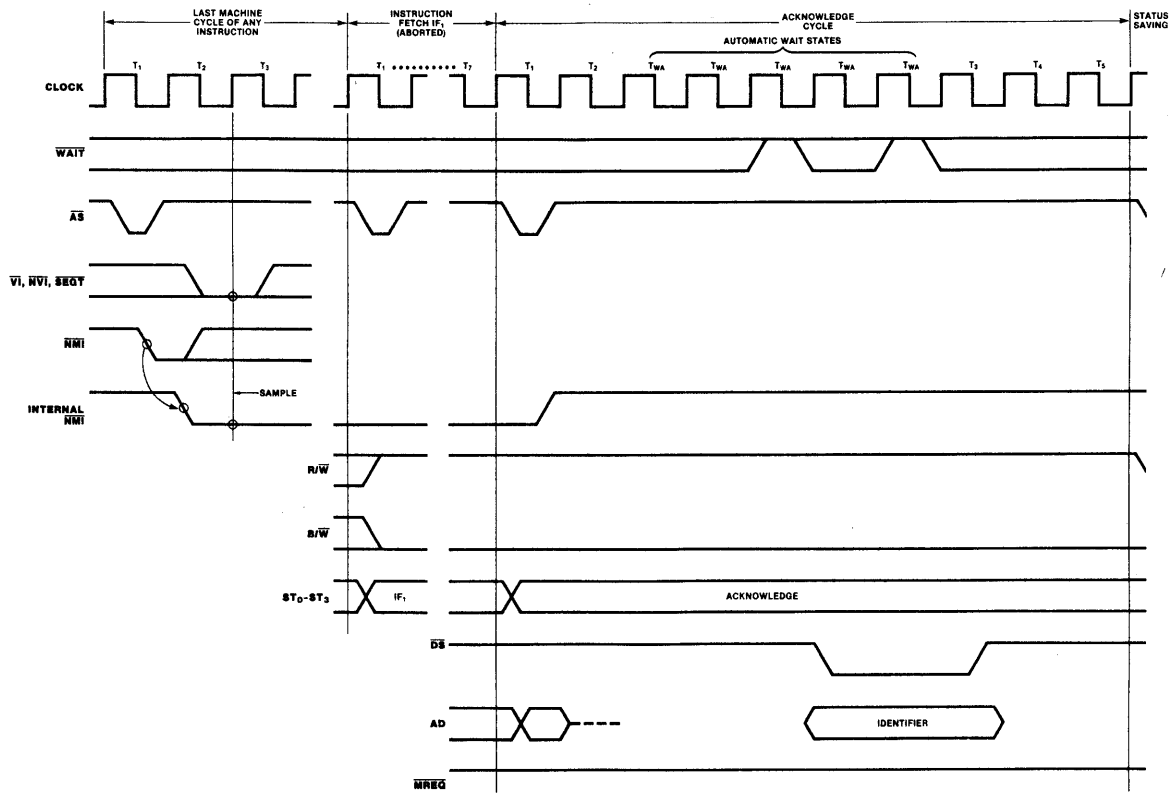


Figure 14. Interrupt and Segment Trap Request/Acknowledge Timing

Status Saving Sequence

The machine cycles following the interrupt acknowledge or segmentation trap acknowledge cycle push the old status information on the system stack (Figure 12) in the following order: the 16-bit program counter; the 7-bit segment number (Z8001 only); the flag control word; and finally the interrupt/trap identifier. Subsequent machine cycles fetch the new program status from the program status area, and then branch to the interrupt/trap service routine.

Bus Request Acknowledge Timing

A Low on the $\overline{BUSREQ}$ input indicates to the CPU that another device is requesting the Address/Data and Control buses. The asynchronous $\overline{BUSREQ}$ input is synchronized at the beginning of any machine cycle (Figure 15). If $\overline{BUSREQ}$ is Low, an internal synchronous $\overline{BUSREQ}$ signal is generated, which–after completion of the current machine cycle–causes the $\overline{BUSACK}$ output to go Low and all bus outputs to go into the high-impedance state. The

Bus Request/ Acknowledge (Continued)

requesting device-typically a DMA-can then control the bus.

When $\overline{BUSREQ}$ is released, it is synchronized with the rising clock edge and the $\overline{BUSACK}$ output goes High one clock period later, indicating that the CPU will again take control of the bus.

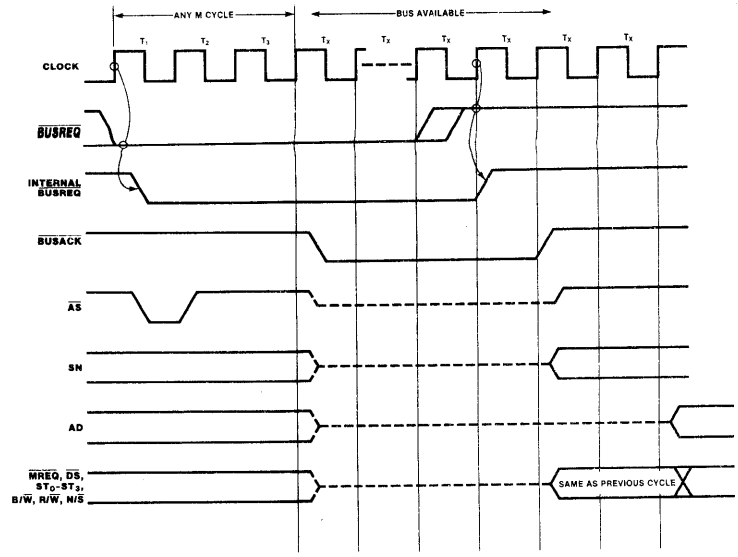


Figure 15. Bus Request/Acknowledge Timing

Stop

The $\overline{STOP}$ input is sampled by the last falling clock edge immediately preceeding any IF_1 cycle (Figure 16) and before the second word of an EPA instruction is fetched. If $\overline{STOP}$ is found Low between the IF_1 cycle, a stream of memory refresh cycles is inserted after T_3, again sampling the $\overline{STOP}$ input on each falling clock edge in the middle of the T_3 states. During the EPA instruction, both EPA instruction words are fetched but any data transfer or subsequent instruction fetch is postponed until $\overline{STOP}$ is sampled High. This refresh operation does not use the refresh prescaler or its divide-by-four clock prescaler; rather, it double-increments the refresh counter every three clock cycles. When $\overline{STOP}$ is found High again, the next refresh cycle is completed, any remaining T states of the IF_1 cycle are then executed and the CPU continues its operation.

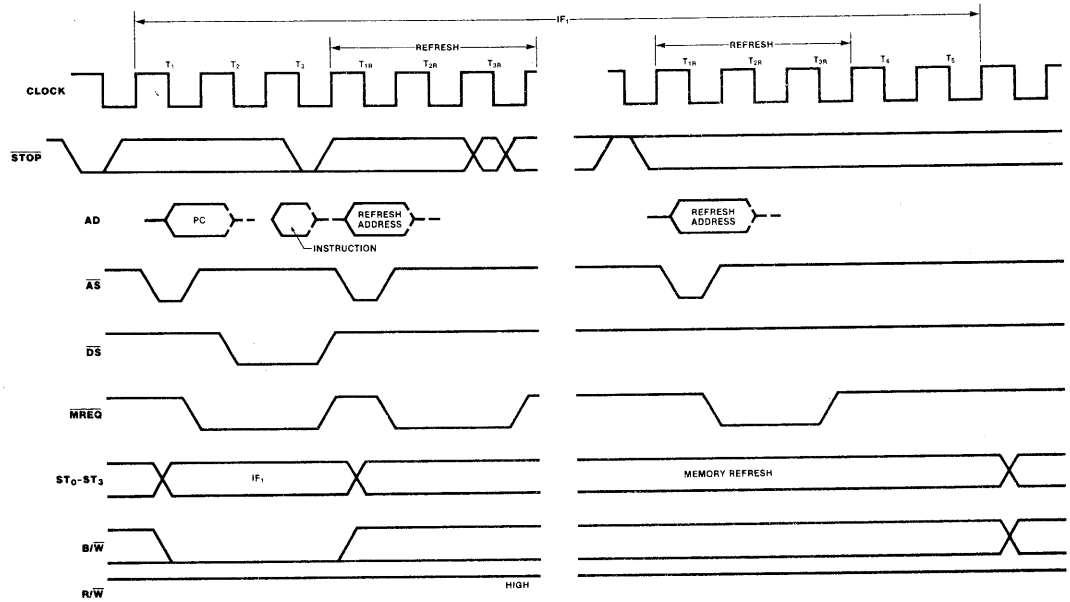


Figure 16. Stop Timing

Certain extended instructions, such as Multiply and Divide, and some special instructions need additional time for the execution of internal operations. In these cases, the CPU goes through a sequence of internal operation machine cycles, each of which is three to eight clock cycles long (Figure 17). This allows fast response to Bus Request and Refresh Request, because bus request or refresh cycles can be inserted at the end of any internal machine cycle.

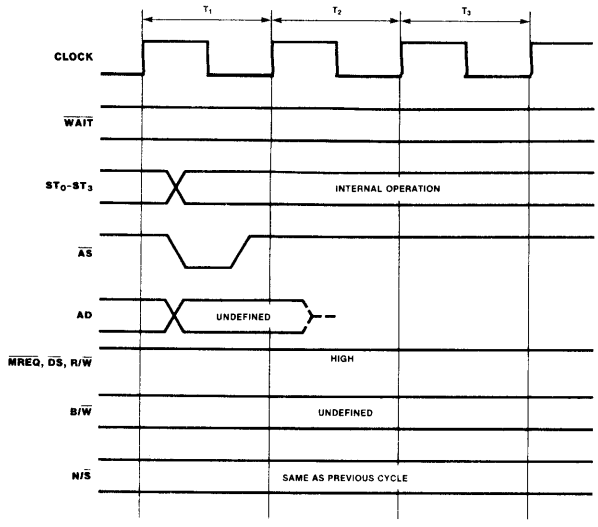


Figure 17. Internal Operation Timing

Memory Refresh

When the 6-bit prescaler in the refresh counter has been decremented to zero, a refresh cycle consisting of three T-states is started as soon as possible (that is, after the next IF_1 cycle or Internal Operation cycle).

The 9-bit refresh counter value is put on the low-order side of the address bus (AD_0-AD_8); AD_9-AD_{15} are undefined (Figure 18). Since the memory is word-organized, A_0 is always Low during refresh and the refresh counter is always incremented by two, thus stepping through 256 consecutive refresh addresses on AD_1-AD_8. Unless disabled, the presettable prescaler runs continuously and the delay in starting a refresh cycle is therefore not cumulative.

While the $\overline{STOP}$ input is Low, a continuous stream of memory refresh cycles, each three T-states long, is executed without using the refresh prescaler.

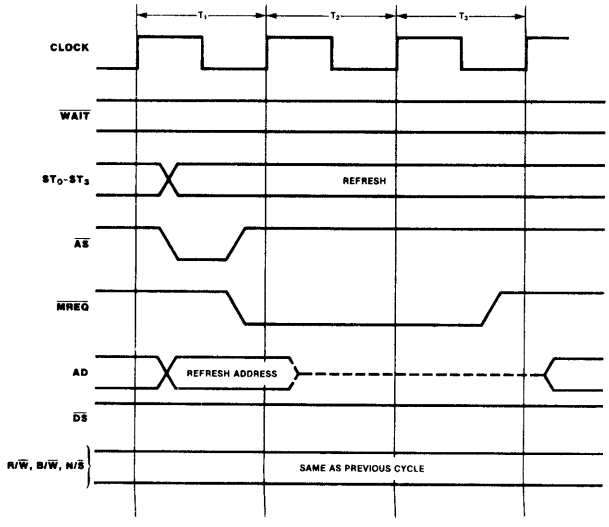


Figure 18. Memory Refresh Timing

Halt

A HALT instruction executes an unlimited number of 3-cycle internal operations, inter-spersed with memory refresh cycles whenever requested. An interrupt, segmentation trap or reset are the only exits from a HALT instruction.

The CPU samples the $\overline{VI}$, $\overline{NVI}$, $\overline{NMI}$ and $\overline{SEGT}$ inputs at the beginning of every T_3 cycle. If an input is found active during two consecutive samples, the subsequent IF_1 cycle is exercised, but ignored, and the normal interrupt acknowledge cycle is started.

Reset

A Low on the $\overline{RESET}$ input causes the following results within five clock cycles (Figure 19):

- AD_0-AD_{15} are 3-stated
- $\overline{AS}$, $\overline{DS}$, MREQ, ST_0-ST_3, $\overline{BUSACK}$ and $\overline{MO}$ are forced High
- SN_0-SN_6 are forced Low
- Refresh is disabled
- R/$\overline{W}$, B/$\overline{W}$ and N/$\overline{S}$ are not affected

When $\overline{RESET}$ has been High for three clock periods, two consecutive memory read cycles are executed in the system mode. In the Z8001, the first cycle reads the flag and control word from location 0002, the next reads the 7-bit program counter segment number from location 0004, the next reads the 16-bit PC offset from location 0006, and the following IF_1 cycle starts the program. In the Z8002, the first cycle reads the flag and control word from location 0002, the next reads the PC from location 0004, and the following IF_1 cycle starts the program.

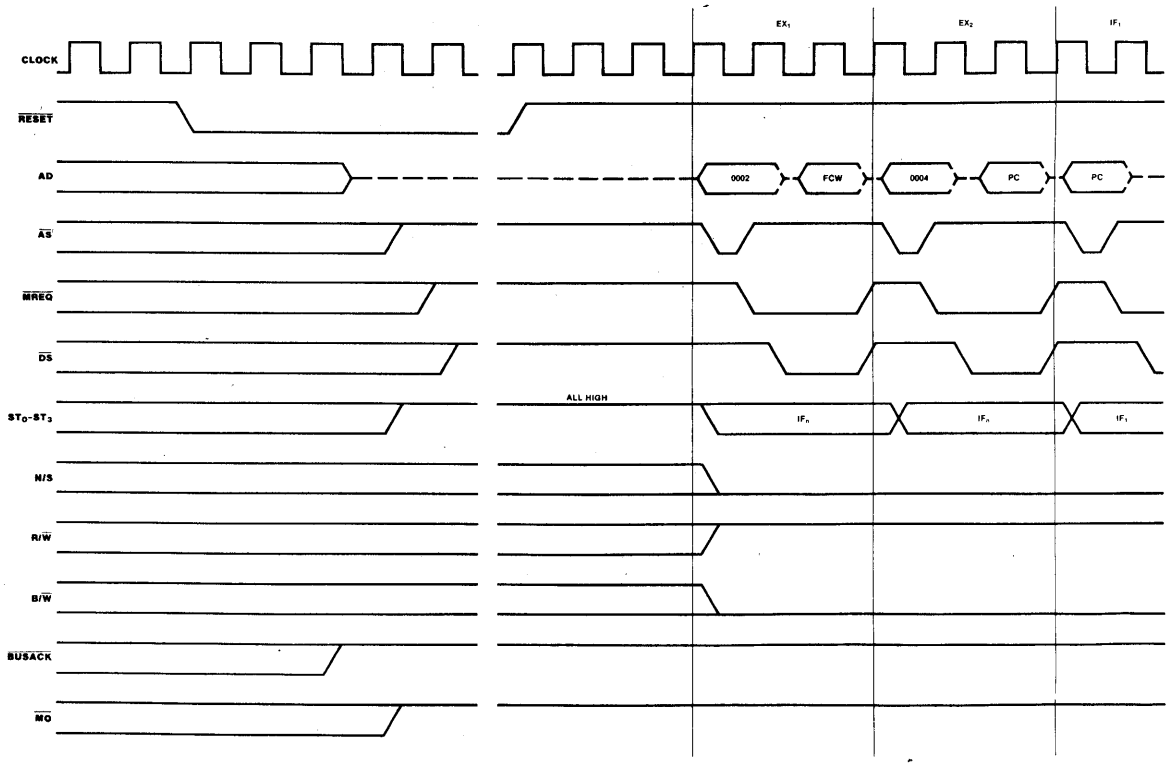
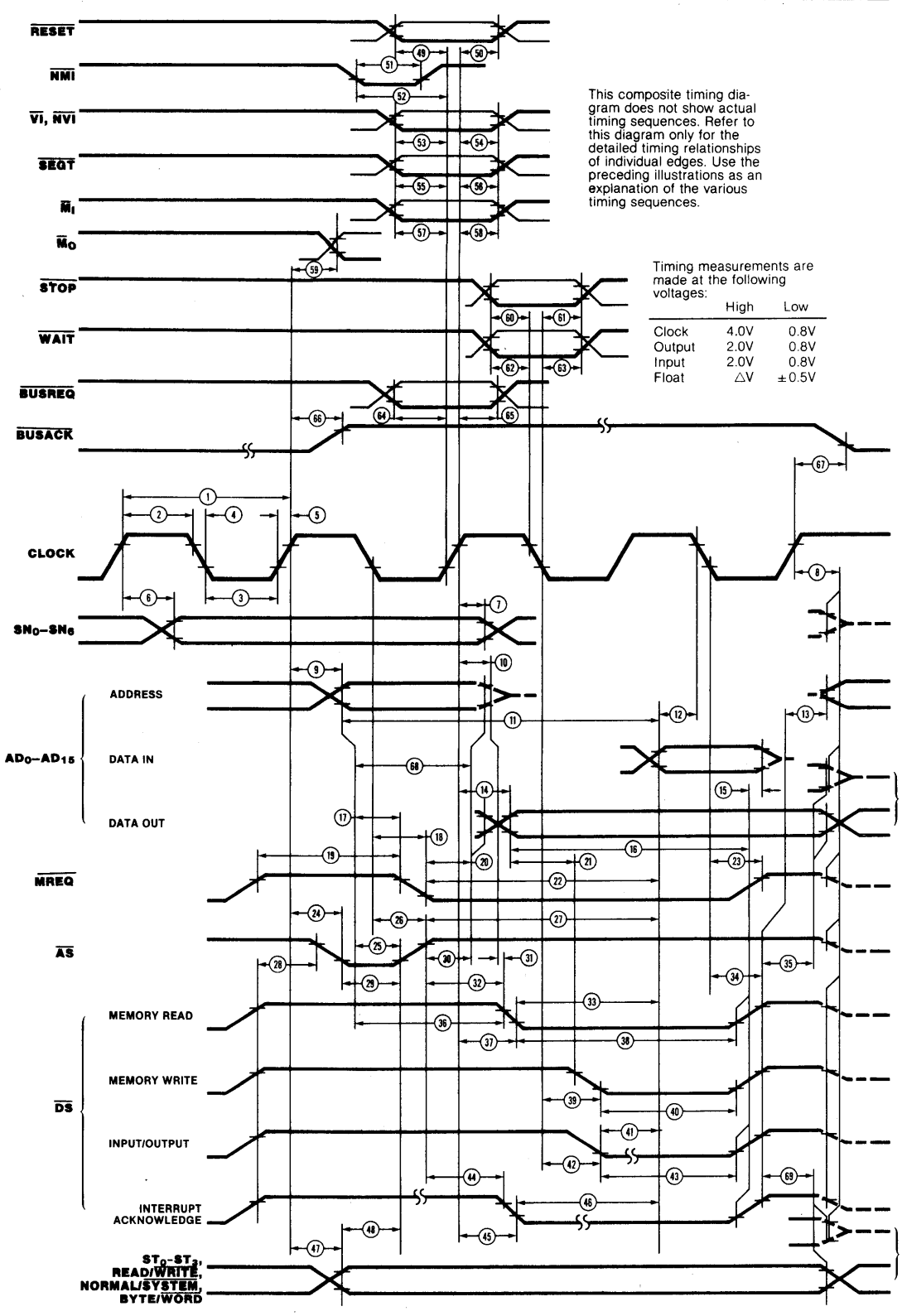


Figure 19. Reset Timing

Composite AC Timing Diagram



This composite timing diagram does not show actual timing sequences. Refer to this diagram only for the detailed timing relationships of individual edges. Use the preceding illustrations as an explanation of the various timing sequences.

Timing measurements are made at the following voltages:

	High	Low
Clock	4.0V	0.8V
Output	2.0V	0.8V
Input	2.0V	0.8V
Float	△V	±0.5V

Number	Symbol	Parameter	Z8001/Z8002 Min	Z8001/Z8002 Max	Z8001A/Z8002A Min	Z8001A/Z8002A Max	Z8001B/Z8002B† Min	Z8001B/Z8002B† Max
1	TcC	Clock Cycle Time	250	2000	165	2000	100	2000
2	TwCh	Clock Width (High)	105	2000	70	2000	40	
3	TwCl	Clock Width (Low)	105	2000	70	2000	40	
4	TfC	Clock Fall Time		20		10		10
5	TrC	Clock Rise Time		20		15		10
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130		110		70
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		10		5	
8	TdC(Bz)	Clock ↑ to Bus Float		65		55		40
9	TdC(A)	Clock ↑ to Address Valid		100		75		50
10	TdC(Az)	Clock ↑ to Address Float		65		55		40
11	TdA(DR)	Address Valid to Read Data Required Valid		475*		305*		180*
12	TsDR(C)	Read Data to Clock ↓ Setup Time	30		20		10	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80*		45*		20*	
14	TdC(DW)	Clock ↑ to Write Data Valid		100		75		50
15	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		0	
16	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	295*		195*		110*	
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55		35*		20*	
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80		70		40
19	TwMRh	$\overline{MREQ}$ Width (High)	210*		135*		80*	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70*		35*		20*	
21	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	55*		35*		15*	
22	TdMR(DR)	$\overline{MREQ}$ ↓ to Read Data Required Valid	375*		230*		140*	
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		80		60		45
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		60		40
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55*		35*		20*	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		80		40
27	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid	360*		220*		140*	
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70*		35*		15*	
29	TwAS	$\overline{AS}$ Width (Low)	85*		55*		30*	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	70*		45*		20*	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0		0	
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	80*		55*		30*	
33	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid	205*		130*		70*	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65		45
35	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid	75*		45*		25*	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180*		110*		65*	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		85		60
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275*		185*		110*	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		80		60
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185*		110*		75*	
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid	330*		210*		120*	
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		90		60
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410*		255*		160*	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065*		690*		410*	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		85		65
46	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Required Delay	455*		295*		165*	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		85		60
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	50*		30*		10*	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		70		50	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		0		0	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		70		50	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		70		50	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		50		40	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	20		20		10	
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		55		40	
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		0		0	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		140		80	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		0		0	
59	TdC(MO)	Clock ↑ to MO Delay		120		85		70
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		100		50	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		0		0	
62	TsW(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		30		20	
63	ThW(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		10		5	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	90		80		60	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		10		5	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		100		75		60
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		100		75		60
68	TwA	Address Valid Width	150*		95*		50*	
69	TdDS(S)	$\overline{DS}$ ↑ to STATUS Not Valid	80*		55*		30*	

*Clock-cycle-time-dependent characteristics. See table on following page.

† Units in nanoseconds (ns). All timings are preliminary.

Number	Symbol	Z8001/Z8002 Equation	Z8001A/Z8002A Equation	Z8001B/Z8002B Equation
11	TdA(DR)	2TcC + TwCh - 130 ns	2TcC + TwCh - 95 ns	2TcC + TwCh - 60 ns
13	TdDS(A)	TwCl - 25 ns	TWCl - 25 ns	TwCl - 20 ns
16	TdDW(DS)	TcC + TwCh - 60 ns	TcC + TwCh - 40	TcC + TwCh - 30 ns
17	TdA(MR)	TwCh - 50 ns	TwCh - 35 ns	TwCh - 20 ns
19	TwMRh	TcC - 40 ns	TcC - 30 ns	TcC - 20 ns
20	TdMR(A)	TwCl - 35 ns	TwCl - 35 ns	TwCl - 20 ns
21	TdDW(DSW)	TwCh - 50 ns	TwCh - 35 ns	TwCh - 25 ns
22	TdMR(DR)	2TcC - 130 ns	2TcC - 100 ns	2TcC - 60 ns
25	TdA(AS)	TwCh - 50 ns	TwCh - 35 ns	TwCh - 20 ns
27	TdAS(DR)	2TcC - 140 ns	2TcC - 110 ns	2TcC - 60 ns
28	TdDS(AS)	TwCl - 35 ns	TwCl - 35 ns	TwCl - 25 ns
29	TwAS	TwCh - 20 ns	TwCh - 15 ns	TwCh - 10 ns
30	TdAS(A)	TwCl - 35 ns	TwCl - 25 ns	TwCl - 20 ns
32	TdAS(DSR)	TwCl - 25 ns	TwCl - 15 ns	TwCl - 10 ns
33	TdDSR(DR)	TcC + TwCh - 150 ns	TcC + TwCh - 105 ns	TcC + TwCh - 70 ns
35	TdDS(DW)	TwCl - 30 ns	TwCl - 25 ns	TwCl - 15 ns
36	TdA(DSR)	TcC - 70 ns	TcC - 55 ns	TcC - 35 ns
38	TwDSR	TcC + TwCh - 80 ns	TcC + TwCh - 50 ns	TcC + TwCh - 30 ns
40	TwDSW	TcC - 65 ns	TcC - 55 ns	TcC - 25 ns
41	TdDSI(DR)	2TcC - 170 ns	2TcC - 120 ns	2TcC - 80 ns
43	TwDS	2TcC - 90 ns	2TcC - 75 ns	2TcC - 40 ns
44	TdAS(DSA)	4TcC + TwCl - 40 ns	4TcC + TwCl - 40 ns	4TcC + TwCl - 30 ns
46	TdDSA(DR)	2TcC + TwCh - 150 ns	2TcC + TwCh - 105 ns	2TcC + TwCh - 75 ns
48	TdS(AS)	TwCh - 55 ns	TwCh - 40 ns	TwCh - 30 ns
68	TwA	TcC - 90 ns	TcC - 70 ns	TcC - 50 ns
69	TdDS(s)	TwCl - 25 ns	TwCl - 15 ns	TwCl - 10 ns

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

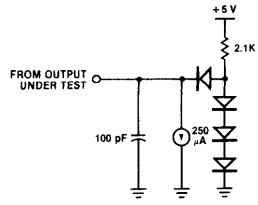
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Test Conditions

The characteristics below apply for the following test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.



All ac parameters assume a total load capacitance (including parasitic capacitances) of 100 pF max, except for parameter 6 (50 pF max). Timing references between two output signals assume a load difference of 50 pF max.

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
$V_{IH\ RESET}$	Input High Voltage on $\overline{RESET}$ pin	2.4	V_{CC} to .3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
$I_{IL\ SEGT}$	Input Leakage on SEGT pin	-100	100	µA	
I_{OL}	Output Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		300	mA	

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8001	CE	4.0 MHz	CPU (segmented, 48-pin)
Z8001	CM	4.0 MHz	Same as above
Z8001	CMB	4.0 MHz	Same as above
Z8001	CS	4.0 MHz	Same as above
Z8001	DE	4.0 MHz	Same as above
Z8001	DS	4.0 MHz	Same as above
Z8001	PE	4.0 MHz	Same as above
Z8001	PS	4.0 MHz	Same as above
Z8001A	CE	6.0 MHz	Same as above
Z8001A	CS	6.0 MHz	Same as above
Z8001A	DE	6.0 MHz	Same as above
Z8001A	DS	6.0 MHz	Same as above
Z8001A	PE	6.0 MHz	Same as above
Z8001A	PS	6.0 MHz	Same as above
Z8002	CE	4.0 MHz	CPU (nonsegmented, 40-pin)
Z8002	CM	4.0 MHz	Same as above
Z8002	CMB	4.0 MHz	Same as above
Z8002	CS	4.0 MHz	Same as above
Z8002	DE	4.0 MHz	Same as above
Z8002	DS	4.0 MHz	CPU (nonsegmented, 40-pin)
Z8002	PE	4.0 MHz	Same as above
Z8002	PS	4.0 MHz	Same as above
Z8002A	CE	6.0 MHz	Same as above
Z8002A	CM	6.0 MHz	Same as above
Z8002A	CMB	6.0 MHz	Same as above
Z8002A	CS	6.0 MHz	Same as above
Z8002A	DE	6.0 MHz	Same as above
Z8002A	DS	6.0 MHz	Same as above
Z8002A	PE	6.0 MHz	Same as above
Z8002A	PS	6.0 MHz	Same as above
Z8002B	CE	10.0 MHz	Same as above
Z8002B	CM	10.0 MHz	Same as above
Z8002B	CMB	10.0 MHz	Same as above
Z8002B	CS	10.0 MHz	Same as above
Z8002B	DE	10.0 MHz	Same as above
Z8002B	DS	10.0 MHz	Same as above
Z8002B	PE	10.0 MHz	Same as above
Z8002B	PS	10.0 MHz	Same as above

*NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8003/4 Z8000™ Z-VMPU Virtual Memory Processing Unit

Zilog

Product Specification

September 1983

FEATURES

- Regular, easy-to-use architecture.
- Instruction set more powerful than many minicomputers.
- Direct addressing capability of up to 8M bytes in each address space.
- Supports implementation of virtual memory systems.
- Eight user-selected addressing modes.
- Wide range of data types including bits, bytes, words, 32-bit long words, and byte and word strings.
- Binary-compatible with Z8001/2 CPUs.
- Separate System and Normal operating modes.
- Sophisticated interrupt structure.
- Resource-sharing capabilities for multiprocessing systems.
- Multi-programming support.
- 32-bit operations, including signed multiply and divide.
- Z-BUS™ compatible.
- Multiple clock rates: 4, 8, or 10 MHz.

GENERAL DESCRIPTION

The Virtual Memory Microprocessor Units (Z8003 and Z8004 Z-VMPUs) accommodate applications that range from the simplest to the most complex.

The Z8003 Z-VMPU uses both segmented and nonsegmented address spaces. It also provides facilities for the implementation of demand segment swapping or a demand paged virtual memory system.

The Z8004 Z-VMPU uses only nonsegmented address spaces. It also provides facilities for the implementation of a demand paged virtual memory system.

Both Z-VMPUs interface with the entire Z8000 Family of support components. Used alone or with Z8000 Family components, the advanced architecture of these LSI Z-VMPUs permits the implementation of systems that have the flexibility and the sophisticated features usually associated with minicomputers or mainframe computers.

The Z8003/4 microprocessors are binary compatible with other Z8000 Family microprocessors. The features that distinguish these microprocessors from the Z8001 and Z8002 microprocessors are the abort capability and the Test and Set status.

An abort request function aids in the implementation of virtual memory systems. The abort function is initiated by memory management circuitry external to the Z-VMPU when an address issued by the Z-VMPU references information (data or instructions) that is not in main memory. After the abort interrupt function, a service routine must bring the page or segment containing the addressed data into main memory. The mainstream program is then restarted at the point of interruption. An abort interrupt differs from a standard interrupt in that the executing instruction is stopped immediately upon detection of the interrupt; this prevents the loss of information needed for a successful restart.

The Test and Set instruction (TSET), in addition to its semaphore test and set function, causes status code 1111 to be placed onto output lines ST_0–ST_3 during the data read bus transaction. It can be used by external circuitry to lock memory to prevent it from being accessed by any other device during the execution of the current TSET instruction.

The architectural features of the Z-VMPU combine to produce a powerful and versatile microprocessor. These features result in the following benefits:

- High-density code
- Efficient compilation of programs
- Support for typical operating system operations
- Complex data structures
- Large-scale virtual memory systems

The Z-VMPU is designed so that a powerful memory management system can be used to improve the utilization of the main memory either as a standard memory or as a virtual memory configuration. Zilog produces Memory Management Units (Z-MMUs) designed for use with the Z8003 Z-VMPU to implement both virtual and nonvirtual memory systems.

The architectural resources of the Z-VMPUs include sixteen l6-bit registers, seven data types (ranging from bits to 32-bit words, and byte and word strings), eight addressing modes, and a powerful instruction set.

A general mechanism has been provided for extending the basic instruction set through the use of external devices called Extended Processing Units (EPUs). In general, an EPU is dedicated to performing complex and time-consuming tasks (such as floating-point arithmetic) so as to unburden the Z-VMPU. Figure 1 shows a simplified block diagram of the Z-VMPU.

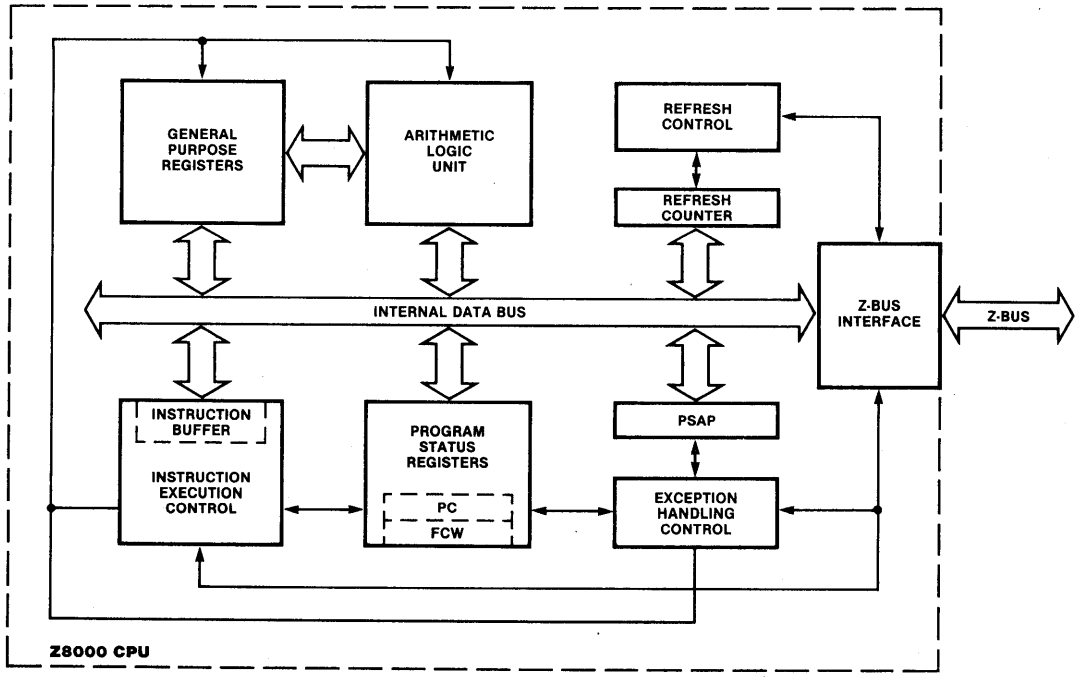


Figure 1. Block Diagram

ARCHITECTURE

General-Purpose Registers

The Z-VMPU is a register-oriented machine that contains sixteen 16-bit general-purpose registers. All general-purpose registers can be used as accumulators and all but one can be used as index registers or memory pointers.

Register flexibility is created by grouping and overlapping multiple registers (Figure 2). For byte operations, the first eight 16-bit registers can be treated as sixteen 8-bit registers. The sixteen 16-bit registers can also be grouped in pairs to form eight 32-bit long-word registers. Similarly, the register set can be grouped in quadruples to form four 64-bit registers.

Stacks. Z-VMPUs can use stacks located anywhere in main memory. Call and Return instructions, as well as interrupts and traps, use an implied stack. Two stack pointers are available, the System Stack Pointer and the Normal Stack Pointer. The two stacks separate operating system (System mode) information from application program (Normal mode) information. The user can manipulate the Stack Pointer with any instruction available for register operations because the Stack Pointer is part of the general-purpose register group.

In the Z8003 Z-VMPU, register pair RR14 is the implied Stack Pointer for segmented operation. Register R14 contains the 7-bit segment number and R15 contains the 16-bit offset. Register R15 is used as the Stack Pointer during nonsegmented operation. Since the Z8004 runs only in the nonsegmented mode, register R15 is used as the Stack Pointer.

Special-Purpose Registers

The Z-VMPUs also provide 16-bit special-purpose registers. These registers include Program Status registers, Program Status Area Pointer register(s), and a Refresh Counter. The configurations of the special-purpose registers for the Z8003 and Z8004 Z-VMPUs are shown in Figure 3.

Program Status Registers. This group of registers consists of the Program Counter (PC) register and the Flag and Control Word (FCW) register. The PC register contains the address of the next instruction to be loaded into the CPU. The low-order byte of the FCW register contains the following flags:

C, Carry flag, is used to indicate that a carry was made out of the high-order bit position of a register used as an accumulator.

Z, Zero flag, is generally used to indicate that the result of an operation was zero.

S, Sign flag, is generally used to indicate that the result of an operation was negative.

P/V, Parity/Overflow flag, is generally used to indicate either even parity (after logical operations on byte operands) or an overflow condition (after arithmetic operations).

D, Decimal-Adjust flag, is used in BCD arithmetic to indicate the type of instruction that was executed (addition or subtraction).

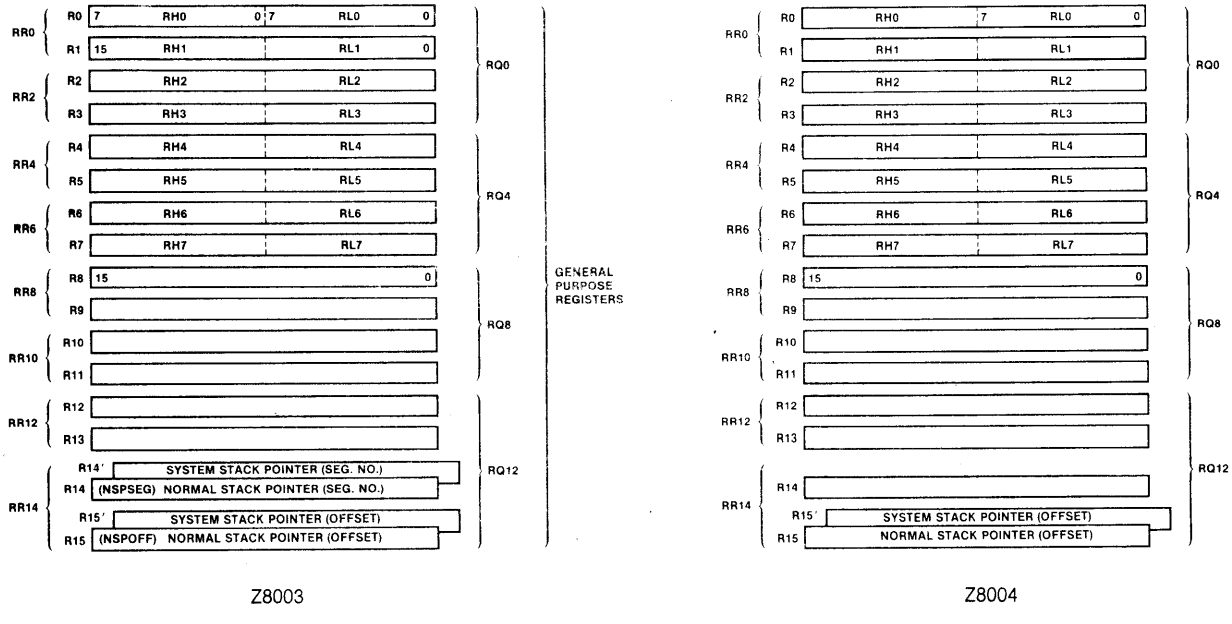


Figure 2. Z-VMPU General-Purpose Registers

H, Half Carry flag, is used to convert the binary result of a previous addition or subtraction into the correct decimal (BCD) result.

The high-order byte of the FCW register contains control bits which are used to control the Z-VMPU operating modes and to enable various types of interrupts. The following control bits are contained in the FCW:

NVIE, Nonvectored Interrupt Enable bit. This bit must be 1 to enable the Z-VMPU to accept non-vectored interrupts.

VIE, Vectored Interrupt Enable bit. This bit must be 1 to enable the Z-VMPU to accept vectored interrupts.

S/$\overline{N}$, System/Normal bit. This bit indicates the current Z-VMPU operating mode. When 0, S/N specifies Normal mode; When 1, S/$\overline{N}$ specifies System mode. The Z-VMPU output N/S represents the complement of this bit.

EPA, Extended Processor Architecture mode bit. This bit, when 1, indicates that the system contains an Extended Processing Unit (EPU) and extended instructions are to be executed by the appropriate EPU. When 0, this bit specifies that extended instructions will be trapped for software emulation.

SEG, Segmentation mode bit (Z8003 only). When 1, this bit specifies that the Z-VMPU is in segmented addressing mode; when 0 it specifies that the Z-VMPU is in the nonsegmented addressing mode.

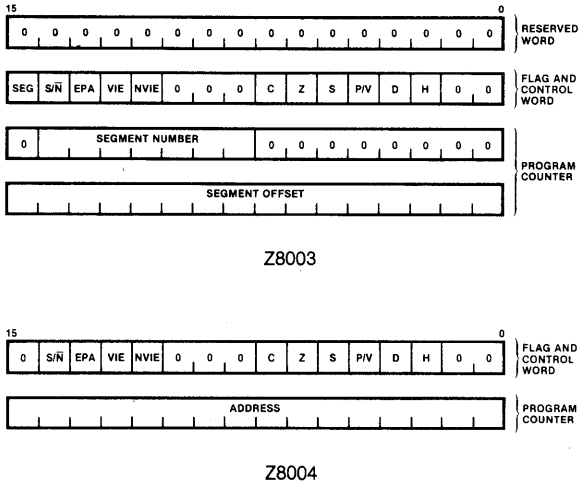


Figure 3. Program Status Registers

Program Status Area Pointer (PSAP) Register. A Program Status Area (PSA) array in main memory is used to store new program status information (i.e., sets of FCW and PC values). Each time an interrupt or trap occurs, the current program status is saved and a new program status is loaded into the status registers from the Program Status Area. The address of the table that contains new program status values is contained in a Program Status Area Pointer (PSAP) register (Figure 4). The low-order byte of the offset address is assumed to be all zeros; therefore, the Program Status Area must start on a 256-byte boundary.

Refresh Register. The Z-VMPU contains a programmable counter that automatically refreshes dynamic memory. The Refresh Counter register consists of a 9-bit row counter, a 6-bit rate counter, and an Enable bit (Figure 5). The 9-bit row counter can address up to 256 rows and is incremented by two each time the rate counter reaches end-of-count. The rate counter determines the time between successive refreshes. It consists of a programmable, 6-bit modulo-n prescaler (n = 1–64), driven at one-fourth the Z-VMPU clock rate. Refresh can be disabled by programming the refresh Enable/Disable bit. If this register is not needed for memory refresh, it can function as an on-board internal timer.

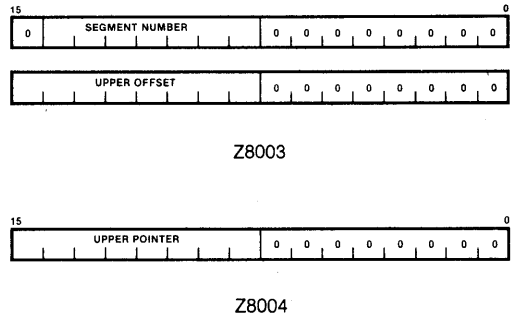


Figure 4. Z8003 Program Status Area Pointer (PSAP) Registers

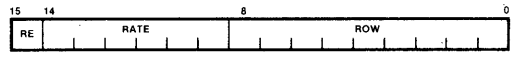


Figure 5. Refresh Register/Counter

SYSTEM AND NORMAL MODES

The Z-VMPUs can run in either System or Normal mode. In System mode, all instructions can be executed and all Z-VMPU control registers can be accessed. This mode is useful in programs that perform operating system functions.

In Normal mode, some instructions, such as the I/0 instructions, cannot be executed. In addition, the Z-VMPU control registers cannot be accessed. This mode is intended for use by application (user) programs.

The use of separate Z-VMPU System and Normal modes promotes the integrity of the system by preventing user programs from having access to the operating system and the control registers. The current operating mode is specified by the S/N bit of the FCW register. The complement of the state of this bit is output by the Z-VMPU on line N/S. Output N/S can be used to separate System and Normal address spaces.

ADDRESS SPACES

Programs and data can be located in the main memory of the computer system or in peripheral devices. In either case, the location of the information must be specified by an address before that information can be accessed. A set of these addresses is called an address space.

The Z-VMPUs support two different types of addresses and thus two categories of address space:

- Memory addresses, which specify locations in main memory.
- I/O addresses, which specify the ports through which peripheral devices are accessed.

Within the two general types of address spaces (memory and I/O), there are several subcategories. Figure 6 shows the address spaces that are available on both types of Z-VMPUs.

The difference between the Z8003 and the Z8004 Z-VMPUs lies not in the number and type of address spaces, but rather in the organization and size of each space. For the Z8003, the memory address space contains 8M bytes of addresses grouped into 128 separate segments. For the Z8004, the memory space is a homogeneous collection of 64K bytes of addresses. In both the Z8003 and the Z8004, each I/O address space contains 32K byte port addresses and 64K word port addresses.

When an address is used to access data, the address spaces can be distinguished by the state of the status lines (ST_0–ST_3) and by the value of the Normal/System line (N/S). The states of the four status lines are determined by the way the address was generated. The value of the N/S output line is the complement of the S/N control bit in the FCW register.

The 23-bit segmented addresses are divided into 7-bit segment identifiers (segment numbers) and 16-bit offsets to address locations relative to the beginning of the specified segment. In hardware, segmented addresses are contained in a register pair or in a long-word memory location. The segment number and offset of an address can be manipulated separately or together by all available word and long word operations.

In an instruction, a segmented address can have one or two representations; long-offset or short-offset. A long-offset address occupies two words, with the first word containing the 7-bit segment number and the second word containing the 16-bit offset. A short-offset address requires only one word, which combines the 7-bit segment number with an 8-bit offset (range 0-256). The short-offset mode allows very dense encoding of addresses and minimizes the need for long addresses to directly access each 8M byte address space.

Nonsegmented addresses are 16 bits and permit access of up to 64K of contiguous byte locations.

The Z8004 operates only in the nonsegmented address mode. The Z8003 can operate in either the segemented or nonsegmented address mode. When the Z8003 is in nonsegmented mode, all address representations assume implicitly the segment number contained in the 7-bit segment number field of the PC.

I/O Addresses

There is a set of I/O instructions that perform 8- or 16-bit transfers between a Z-VMPU and its I/O devices. I/O devices are addressed with 16-bit I/O port addresses. An I/O port address is similar to a memory address; however, the I/O address space is not part of the memory address space. Memory-mapped I/O can be implemented by dedicating memory locations to I/O device registers. Two types of I/O instruction are available: Standard and Special. Each type has its own address space. Special I/O instructions are used for loading and unloading memory management units.

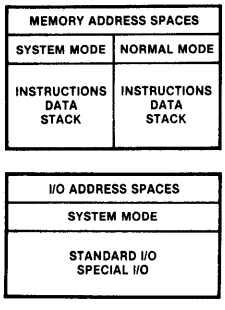


Figure 6. Address Spaces on the Z8003 and Z8004

INSTRUCTION ADDRESSING MODES

The information included in Z-VMPU instructions consists of the function to be performed, the type and size of data elements to be manipulated, and the locations of the data elements. Locations are designated by register addresses, memory addresses, or I/O addresses. The addressing mode of a given instruction defines the method used to compute the address. Addressing modes are explicitly specified or implied by the instruction. Locations are designated using one of the following addressing modes:

- **Register Mode (R).** The data element is located in one of the 16 general-purpose registers or a control register.
- **Immediate Mode (IM).** The data element is located in the instruction.
- **Indirect Register Mode (IR).** The data element can be found in the location whose address is given in a specified register.
- **Direct Address Mode (DA).** The data element can be found in the location whose address is given in the instruction.
- **Index Mode (X).** The data element can be found in the location whose address is the sum of the contents of an index value in a specified register and an address in the instruction.
- **Relative Address Mode (RA).** The data element can be found in the location whose address is the sum of the contents of the Program Counter and a displacement given in the instruction.
- **Base Address Mode (BA).** The data element can be found in the location whose address is the sum of a base address in a specified register and a displacement given in the instruction.
- **Base Index Mode (BX).** The data element can be found in the location whose address is the sum of a base address in one specified register and an index value in a second specified register.

INSTRUCTION SET

Major Groups

The major groups of instructions provided by the Z-VMPU are described in the following paragraphs. A detailed summary of the instructions is presented in Table 3 (located at the back of this document).

Load and Exchange. These instructions move data among registers or between registers and main memory.

Arithmetic. These instructions perform integer arithmetic. The basic instructions (e.g., add, subtract, multiply and divide) in this group use standard two's complement binary format. Support is also provided for implementing BCD arithmetic.

Logical. These instructions perform logical operations (i.e., AND, OR, XOR, and complementation) on the bits of specified operands. The operands can be bytes or words. The Test Long (TESTL) instruction, however, permits logical operations to be performed on 32-bit quantities.

Program Control. These instructions affect the Program Counter, thereby controlling program flow.

Bit Manipulation. These instructions manipulate individual bits in registers or main memory.

Rotate and Shift. These instructions shift and rotate the contents of registers.

Block Transfer and String Manipulation. These instructions perform string comparisons, string translations, and block transfer functions.

Input/Output. These instructions transfer bytes, words, or blocks of data between peripheral devices and the Z-VMPU registers or main memory.

Z-VMPU Control. These instructions modify Z-VMPU control and status registers or perform those functions that do not fit into any of the preceding instruction groups.

Extended. These instructions perform Extended Processor Unit (EPU) internal operations, data transfers between memory and EPU, data transfers between EPU and the Z-VMPU, and data transfers between EPU flag registers and the Z-VMPU Flag And Control Word (FCW).

Processor Flags

The processor flags contained by the program status registers provide a link between sequentially executed instructions. The link is provided in the sense that the result of executing one instruction may alter one or more flags. The new flag values (states) can then be used to determine the operation of a subsequent instruction (typically a conditional jump instruction). The following six flags are available for use by the programmer and the processor:

- Carry (C)
- Zero (Z)
- Sign (S)
- Parity/Overflow (P/V)
- Decimal-Adjust (D)
- Half Carry (H)

Table 1. Condition Codes

Code	Meaning	Flag Settings	CC Field Binary	CC Field Hex
F	Always false	—	0000	0
T	Always true	—	1000	8
Z	Zero	Z = 1	0110	6
NZ	Not zero	Z = 0	1110	E
C	Carry	C = 1	0111	7
NC	No carry	C = 0	1111	F
PL	Plus	S = 0	1101	D
MI	Minus	S = 1	0101	5
NE	Not equal	Z = 0	1110	E
EQ	Equal	Z = 1	0110	6
OV	Overflow	P/V = 1	0100	4
NOV	No overflow	P/V = 0	1100	C
PE	Parity is even	P/V = 1	0100	4
PO	Parity is odd	P/V = 0	1100	C
GE	Greater than or equal (signed)	(S XOR P/V) = 0	1001	9
LT	Less than (signed)	(S XOR P/V) = 1	0001	1
GT	Greater than (signed)	[Z OR (S XOR P/V)] = 0	1010	A
LE	Less than or equal (signed)	[Z OR (S XOR P/V)] = 1	0010	2
UGE	Unsigned greater than or equal	C = 0	1111	F
ULT	Unsigned less than	C = 1	0111	7
UGT	Unsigned greater than	[(C = 0) AND (Z = 0)] = 1	1011	B
ULE	Unsigned less than or equal	(C OR Z) = 1	0011	3

Note: Some condition codes have identical flag settings and binary fields in the instruction, i.e., Z = EQ, NZ = NE, C = ULT, NC = UGE, OV = PE, NOV = PO.

Condition Codes

Flags C, Z, S, and P/V are used to control the operation of conditional instructions (such as Conditional Jump). The operations performed by this type of instruction depend on whether or not a specified Boolean condition exists on the four flags. Sixteen functions of the flag settings found to be frequently used are encoded in a 4-bit condition code (CC) field, which forms a part of all conditional instructions. These 16 codes are described in Table 1.

MULTI-MICROPROCESSOR RESOURCE CONTROL

The Z8003 and Z8004 Z-VMPUs include both hardware and software support for controlling access to shared resources in multi-microprocessor systems. Z-VMPU pins $\overline{MI}$ (Multi-Micro In) and $\overline{MO}$ (Multi-Micro Out) and instructions MSET (Set $\overline{MO}$), MREQ (access request), MBIT (Test $\overline{MI}$), and MRES (reset $\overline{MO}$) can be used to form a prioritized resource access control system. Such a system would, for a Z-VMPU, 1) issue requests for access to a shared resource, 2) test the access status for the resource (available/not available) and 3) when access is granted, exclude all other Z-VMPUs in the system from the resource until use of the resource is complete.

TEST AND SET INSTRUCTION (TSET)

The TSET instruction implements synchronization mechanisms in multiprogramming and multiprocessing environments. TSET tests and sets semaphores that control access to shared resources. The testing and setting of a semaphore requires the semaphore to be read from memory, modified, then written back into the same memory location. To prevent other processors from requesting access to a resource during a test and set process, status code 1111 is placed onto status lines ST_0–ST_3 during the data read transaction to specify that an uninterruptable memory operation is taking place. Status code 1111 is particularly useful in a multiple microprocessor environment to permit external circuitry to preclude memory access by another device between the read transaction and the write transaction of the test and set operation. Request input BUSREQ is also disabled during a test and set operation to ensure that the test and set operation is not interrupted; this action is useful in a single-processor system.

EXTENDED PROCESSING ARCHITECTURE

The Z-VMPU has an Extended Processing Architecture (EPA) facility which extends the basic functions of the Z-VMPU by using external devices called Extended Processing Units (EPUs). A special set of extended instructions controls the operations to be performed by each EPU. When a Z-VMPU encounters an extended instruction, it either traps the instruction, or it performs the data transfer portion of the instruction. The data manipulation portion of the instruction is executed by the involved EPU. Whether the Z-VMPU traps or transfers data depends on the setting of an EPA bit in its Flag and Control Word (FCW) status register.

EXCEPTIONS

The Z8003 and Z8004 Z-VMPUs support four types of exceptions (conditions that alter the normal flow of program execution): interrupts, traps, instruction aborts, and reset.

Interrupt and Trap Structure

The Z8003 and Z8004 Z-VMPUs have a flexible and powerful interrupt and trap structure. Interrupts are external events requiring Z-VMPU attention and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions.

Both Z8003 and Z8004 Z-VMPUs support three interrupts: nonmaskable ($\overline{NMI}$), vectored ($\overline{VI}$), and nonvectored ($\overline{NVI}$).

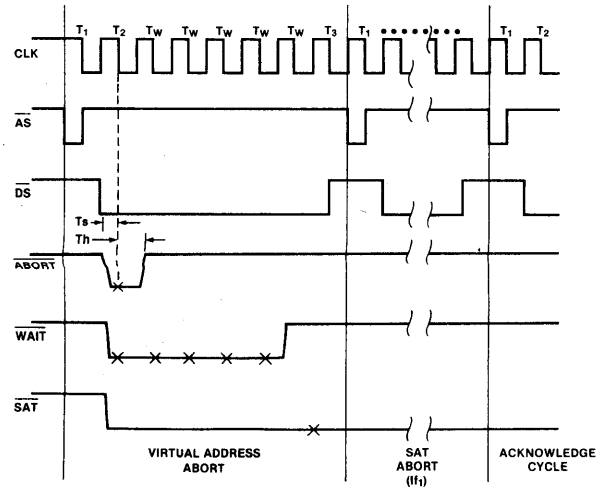
Both Z-VMPUs support several types of traps: System Call, EPU instruction, and privileged instruction. In addition, the Z8003 supports a Segment/ Address Translation ($\overline{SAT}$) trap. Of the above traps, only the last is initiated by external events. Such events are normally generated by a memory management system. The remaining traps occur when instructions limited to the System mode are used in the Normal mode, when a System Call instruction is executed, or when an EPA instruction is encountered.

The descending order of priority for traps and interrupts is: internal traps, nonmaskable interrupts, segment/address translation traps, vectored interrupts, and nonvectored interrupts.

When an interrupt or trap occurs, the current program status information is automatically pushed onto the System stack. The new program status is then automatically loaded into the Program Status registers from the Program Status Area in System program memory. This area of memory is identified by the Program Status Area Pointer (PSAP).

Instruction Abort Function

The Z-VMPU monitors its $\overline{ABORT}$ input during each bus transaction it generates. The timing for an Instruction Abort operation is shown in Figure 7. If the $\overline{ABORT}$ input is asserted during clock cycle T_2 of a memory access, the currently executing instruction is automatically aborted. If no abort is indicated but input $\overline{WAIT}$ is asserted, input $\overline{ABORT}$ is also tested during each wait cycle (T_W). When an Instruction Abort condition is indicated ($\overline{ABORT}$ is asserted) the $\overline{WAIT}$ input must also be asserted for five cycles to permit the Z-VMPU internal control mechanism to abort the current instruction. When the $\overline{WAIT}$ input is deasserted, the Z-VMPU acknowledges any pending interrupt request. Therefore, the memory management circuitry that caused the interrupt to be aborted should also request an interrupt to the software routine that restores the Z-VMPU registers and the main memory so that the aborted instruction can be reissued.



NOTE: * = Clock Sample Points

Figure 7. Instruction Abort Timing

VIRTUAL MEMORY SYSTEMS

Virtual memory systems permit programs to reference an address space that exceeds the main (physical) memory. In virtual memory systems, high-speed main memory is supported by medium- and low-speed storage devices (secondary memory) such as hard disks or floppy disks. When a Z-VMPU in a virtual system issues an address that references information not in main memory, a software swap operation must be initiated. This swap retrieves the block containing the referenced location, loads it into main memory, and restarts the aborted mainstream program at the point of interruption. The swap operation is transparent to the user and to the executing program; therefore, the system appears to have a memory that is not constrained by physical size. The maximum size of a virtual memory is determined by the address structure used and by the capabilities of the system memory management hardware and software.

Segmented and Paged Virtual Memories

External circuitry can be used to implement either a segmented virtual memory or a paged virtual memory. In a segmented virtual memory, information is transferred between main memory and secondary storage devices on a segment-by-segment basis. The Z8003 Z-VMPU permits use of variable-length segments of up to 64K bytes.

In a paged virtual memory system, each segment is divided into fixed-size pages (standard size is 2048 bytes). Main memory is divided into page "frames." Information is then transferred between main memory and the secondary storage devices on a page-by-page basis. The Z8003 Z-VMPU can support both segmented or paged virtual memory systems. The Z8004 supports only the paged virtual memory approach.

External Hardware Support

The detection of a logical address that references a location outside main memory (i.e., an addressing fault) and the initiation of the required swap operation must be performed by memory management circuitry external to the Z-VMPU.

A swap operation is started by the initiation of a Segment/Address Translation ($\overline{SAT}$) trap request function in the Z-VMPU. Since the Z8004 does not have a $\overline{SAT}$ input, one of the $\overline{NMI}$, $\overline{VI}$ or $\overline{NVI}$ inputs must be used instead. Low levels on Z-VMPU inputs $\overline{ABORT}$, $\overline{SAT}$ and $\overline{WAIT}$ initiate $\overline{SAT}$ requests.

These inputs are sampled at the falling clock of the second clock cycle of a bus transaction. Input $\overline{WAIT}$ must be asserted for at least five clock cycles. Input $\overline{ABORT}$ must be deasserted on or before the rising edge of the $\overline{WAIT}$ signal. The same timing can be used for both $\overline{WAIT}$ and $\overline{ABORT}$. Input $\overline{SAT}$ should be asserted until the trap acknowledge bus transaction is indicated by Z8003 Z-VMPU status code 0100.

External circuitry is needed to record the information for instruction restart. The following assumptions about the operating system must also be true:

- The fault handler does not generate a fault until all critical data is saved.
- Accessing the System stack never causes a fault. (Either the segment is in memory or a memory management mechanism warns of a potential stack overflow.)
- I/O buffers are always in main memory, so I/O instructions never cause a fault.

- The Program Status Area is always in main memory.

The following information must be saved by external circuitry to restart the instruction interrupted by the addressing fault:

- The value of the Program Counter during the initial instruction fetch cycle (cycle identified by status code 1101).
- The address that caused the fault.
- The code that was on the status lines during the aborted cycle.
- For paged memories, the number of successful data accesses made by the instruction.

Software Support

The software required for virtual memory operation normally consists of a fault handler and a restart routine. The fault handler is started during each Z-VMPU abort request operation. The fault handler is responsible for saving information about the aborted instruction and for the initiation of a request which brings the segment (or page) containing the referenced location in main memory. The state of the aborted program (Flag and Control Word (FCW), Program Counter (PC), and the register file must be saved and another process dispatched while the missing segment (or page) is being fetched from secondary memory.

When the page or segment containing the referenced location is loaded into main memory, an instruction restart routine must be executed. This instruction restart routine must restore the operating environment that existed when the instruction/program abort was initiated. This routine must establish the PC value that points to the aborted instruction. It must also decode the instruction's opcode to determine whether or not any of the Z-VMPU's registers were modified before the instruction execution cycle in which the abort occurred. If registers were modified, the instruction restart routine must return these registers to a state in which the restarted instruction behaves as if no abort had occurred. The flow chart in Figure 8 illustrates a possible control sequence for a software restart routine. The instructions requiring remodification of system registers and the manner in which these registers must be modified depend upon the type (segmented or paged) of virtual memory system implemented.

BUS TRANSACTIONS

Status Outputs

The Z-VMPUs provide output that specifies the type of transaction on the Address/Data bus. Output line R/$\overline{W}$ specifies whether a read or write operation is involved. Output line B/$\overline{W}$ specifies whether the transaction involves byte or word data. Output line N/$\overline{S}$ specifies the mode of operation, Normal or System. In addition to these lines, output lines ST_0–ST_3 encode additional characteristics of the current bus transaction. These lines can present any of sixteen 4-bit status codes which define specific characteristics of the current bus transaction. The available status codes are listed and defined in Table 2.

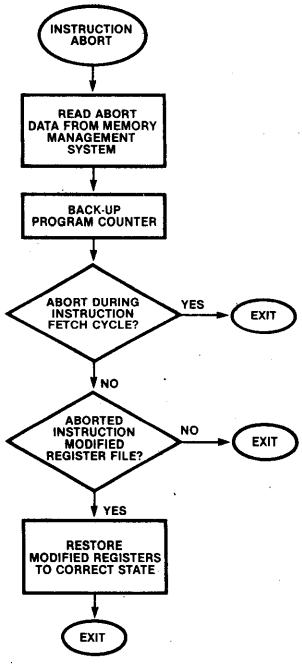


Figure 8. Flow Chart of an Instruction Restart Routine

Table 2. Status Codes

ST_3-ST_0 Binary	Definition
0 0 0 0	Internal Operation
0 0 0 1	Memory Refresh
0 0 1 0	I/O Reference
0 0 1 1	Special I/O Reference (e.g., to an MMU)
0 1 0 0	Segment/Address Translation Trap Acknowledge
0 1 0 1	Nonmaskable Interrupt Acknowledge
0 1 1 0	Nonvectored Interrupt Acknowledge
0 1 1 1	Vectored Interrupt Acknowledge
1 0 0 0	Data Memory Request
1 0 0 1	Stack Memory Request
1 0 1 0	Data Memory Request (Extended Processing Architecture)
1 0 1 1	Stack Memory Request, (Extended Processing Architecture)
1 1 0 0	Instruction Space Access
1 1 0 1	Instruction Fetch, First Word
1 1 1 0	Extended Processing Unit—Z-VMPU Transfer
1 1 1 1	Bus Lock, Data Memory Request

Memory Read and Write

Memory read and instruction fetch cycles are identical, except for the status code on the ST_0–ST_3 outputs. Memory write is similar to memory read except for the R/$\overline{W}$ status and the timing of $\overline{DS}$ and data valid true. During a memory cycle, a 16-bit offset address is placed on the AD_0–AD_{15} outputs early in the first clock period (Figure 9). In the Z8003, a 7-bit segment number is also output on SN_0–SN_6 one clock period earlier than the 16-bit address offset. Issuing the segment number early minimizes address translation overhead by enabling the memory management circuitry to overlap its operations with the Z-VMPU instruction execution cycle.

A valid address is indicated by the rising edge of Address Strobe ($\overline{AS}$). Status and mode information becomes valid early in the memory access cycle and remains stable throughout it. The access cycle can be extended in length by the addition of wait cycles.

The Read/Write line (R/$\overline{W}$) indicates the direction of the data transfer. R/$\overline{W}$ is High for transfers to the Z-VMPU. R/$\overline{W}$ is Low for transfers from the Z-VMPU.

Word data (B/$\overline{W}$ is Low) to or from the Z-VMPU is transmitted on lines AD_0–AD_{15}. Byte data to the Z-VMPU is transmitted in AD_{10}–AD_7, from odd addresses ($AD_0 = 1$) and in AD_8–AD_{15} from even addresses ($AD_0 = 0$). Byte data from the Z-VMPU is replicated in AD_0–AD_7 and AD_8–AD_{15}, regardless of address.

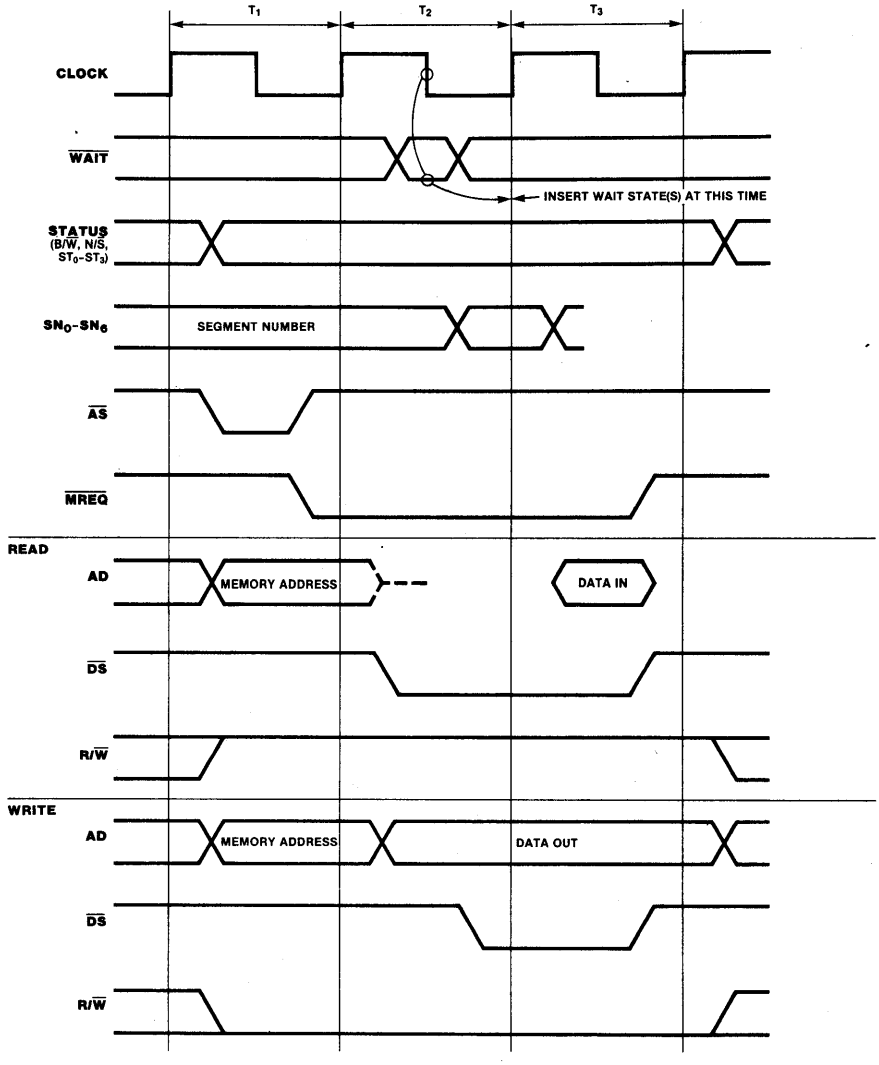


Figure 9. Memory Read and Write Timing

I/O Transactions

I/O transactions, which are generated by the execution of I/O instructions, move data to or from peripherals or Z-VMPU support devices. As shown in the timing diagram presented in Figure 10, I/O transactions have a minimum length of four clock cycles; wait cycles can be added to lengthen transaction periods to meet the needs of slow peripherals. Status line outputs indicate whether access is to the Standard I/O (0010) or Special I/O (0011) address spaces.

I/O transactions are always performed with the Z-VMPU in System mode (N/$\overline{S}$ = Low). The rising edge of $\overline{AS}$ indicates that a valid address is present on lines AD_0–AD_{15}. Since the I/O address is always 16 bits long, the segment number lines in Z8003 are undefined.

For byte transfers (B/$\overline{W}$ = High) in Standard I/O space, addresses must be odd; for byte transfers in Special I/O space, addresses must be even.

Word data (B/$\overline{W}$ = Low) to or from the CPU is transmitted on AD_0–AD_{15}. Byte data (B/$\overline{W}$ = High) is transmitted on AD_0–AD_{15} for Special I/O. This allows peripheral devices or CPU support devices to attach to only eight of the 16 AD_0–AD_{16} lines. The Read/Write line (R/$\overline{W}$) indicates the direction of the data transfer: peripheral-to-CPU (Read: R/$\overline{W}$ = High) or CPU-to-peripheral (Write: R/$\overline{W}$ = Low).

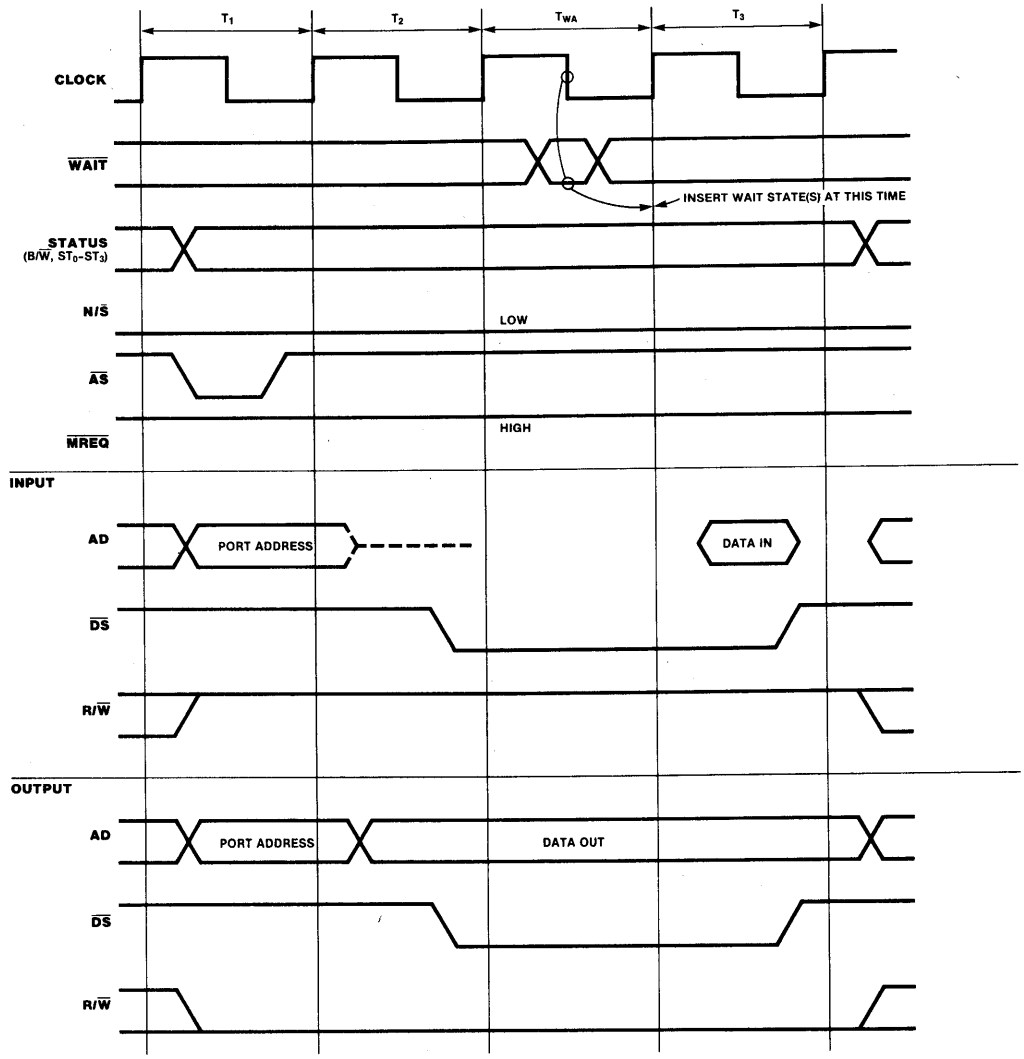


Figure 10. Input/Output Transaction

Wait Add-On Cycles

As shown in Figures 9 and 10, the $\overline{WAIT}$ input line is sampled on a falling edge of CLK one cycle before data is sampled ($\overline{DS}$ is Low for a read or write operation). If the $\overline{WAIT}$ input line is Low when sampled, another cycle is added to the transaction before data is sampled or $\overline{DS}$ is deasserted (goes High). During an added wait cycle, input $\overline{WAIT}$ is sampled again on the falling clock edge; if it is Low, another wait cycle is added to the transaction. This use of the $\overline{WAIT}$ input permits transactions to be extended arbitrarily to accommodate, for example, slow memories or I/O devices that are not yet ready for data transfer.

Memory Refresh Timing

When the 6-bit prescaler in the refresh counter has been decremented to zero, a refresh cycle is started (Figure 11). The 9-bit refresh counter value is put on AD_0–AD_8; lines AD_9–AD_{15} are undefined. Unless disabled, the presettable prescaler runs continuously, therefore any delay in starting a refresh cycle is not cumulative.

While the $\overline{STOP}$ input is Low, a continous stream of memory refresh cycles is executed without using the refresh prescaler. The refresh count, however, is incremented.

Internal Operation Timing

Certain instructions, such as multiply and divide, need additional time to execute internal operations. In these cases, the Z-VMPU goes through a sequence of internal operation machine cycles, each three to eight clock cycles long (Figure 12). This allows fast response to bus and refresh requests because a bus request or a refresh cycle can be inserted at the end of any internal machine cycle.

Although the address outputs during clock cycle T_1 are undefined, Address Strobe ($\overline{AS}$) is generated to satisfy the requirements of Z-BUS-compatible peripherals and self-refresh dynamic memories.

Reset Function

A Low on the $\overline{RESET}$ input causes the following results within five clock cycles (Figure 13):

1. AD_0–AD_{15} are 3-stated.
2. $\overline{AS}$, $\overline{DS}$, $\overline{MREQ}$, $\overline{BUSACK}$, $\overline{MO}$, and ST_0–ST_3 are forced High.
3. SN_0–SN_6 are forced Low.
4. Refresh is disabled.
5. $R/\overline{W}$, $B/\overline{W}$ and $N/\overline{S}$ are undefined.

When $\overline{RESET}$ is again High, the Z8003 Z-VMPU executes three memory read cycles in a System mode of operation. During these three word read cycles, the Z-VMPU reads, in sequence, the following information from segment 0:

1. The flag and control word (FCW) from offset location 0002.
2. The Program Counter segment number from location 0004 and offset from location 0006.

In the Z8004 Z-VMPU, only two read cycles are performed. During the first cycle, the FCW is read from location 0002. During the second cycle, the 16-bit PC value is read from location 0004. The program is started during the following machine cycle.

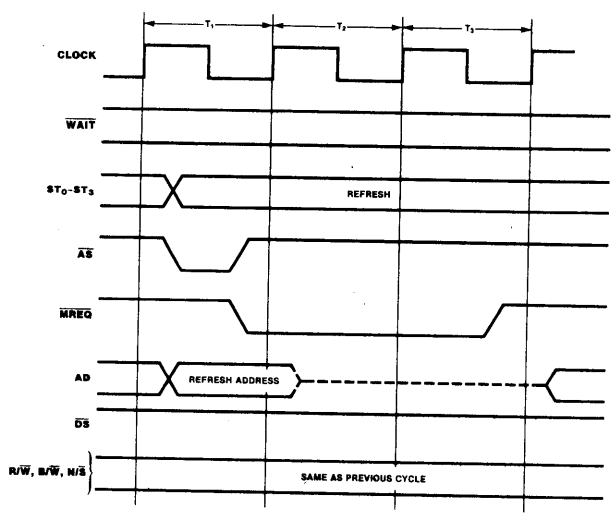


Figure 11. Memory Refresh Timing

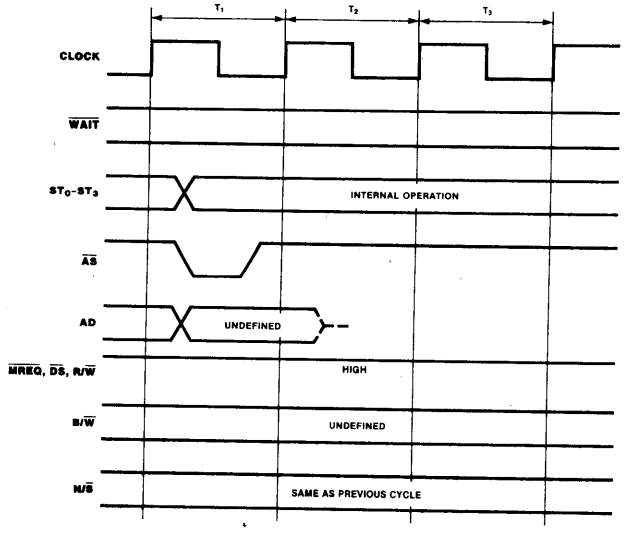


Figure 12. Internal Operating Timing

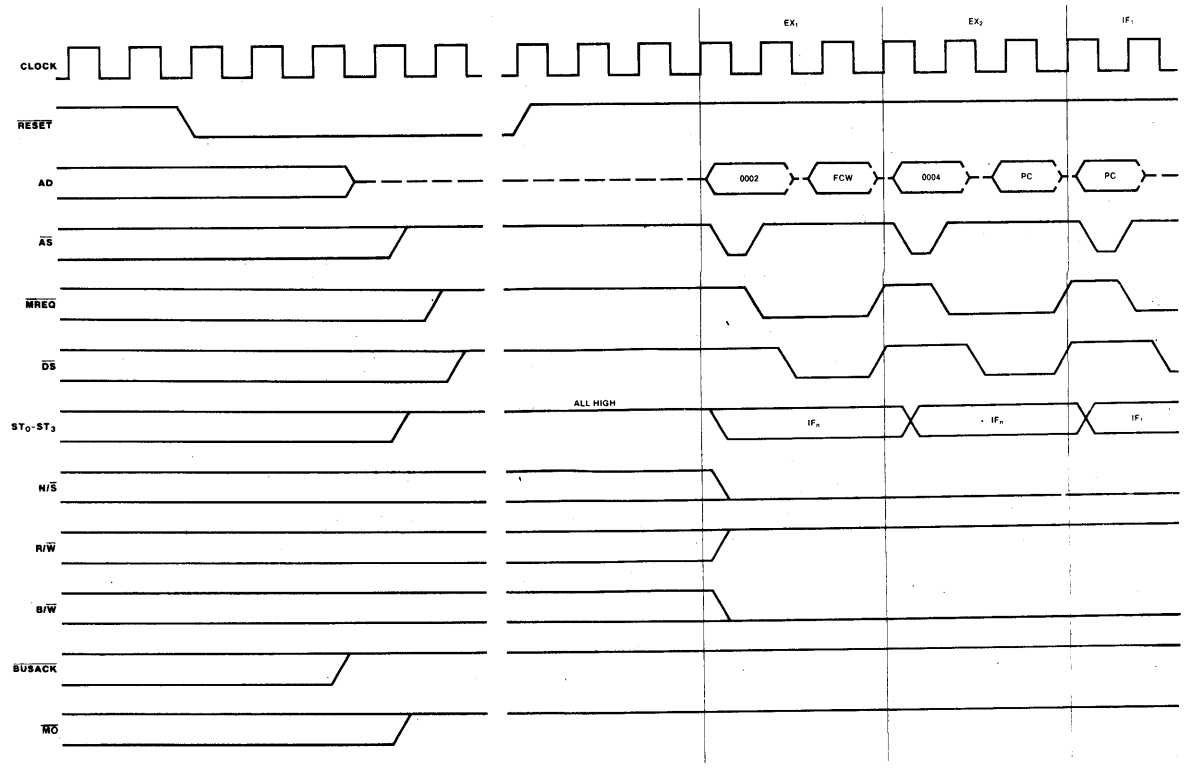


Figure 13. Reset Timing

BUS REQUEST, INTERRUPT AND ACKNOWLEDGE

A Low on the $\overline{BUSREQ}$ input indicates to the Z-VMPU that another device is requesting the address/data and control lines. The asynchronous $\overline{BUSREQ}$ input is synchronized at the beginning of any machine cycle (Figure 14). If $\overline{BUSREQ}$ is Low, an internal synchronous $\overline{BUSREQ}$ signal is generated, which, after completion of the current machine cycle, causes the $\overline{BUSACK}$ output to go Low and all bus outputs to go into the high-impedance state. The requesting device (typically a DMA) can then control the bus.

When $\overline{BUSREQ}$ is released, it is synchronized with the rising clock edge. The $\overline{BUSACK}$ output goes High one clock period later to indicate that the Z-VMPU will take control of the bus.

Interrupt and Segment/Address Translation Trap Request and Acknowledge

Any High-to-Low transition on the Z-VMPU's $\overline{NMI}$ input (Figure 15) is asynchronously edge-detected and sets the internal $\overline{NMI}$ latch. The $\overline{VI}$, $\overline{NVI}$, and $\overline{SAT}$ inputs, as well as the state of the internal $\overline{NMI}$ latch, are sampled at the beginning of T_3.

In response to an interrupt or trap, the subsequent IF_1 cycle is exercised. The Program Counter, however, is not updated, but the System Stack Pointer is decremented in preparation for storing status information on the System stack.

The next machine cycle is the interrupt acknowledge cycle. This cycle has five automatic wait states, and additional wait states are possible.

After the last wait state, the Z-VMPU reads the information on AD_0–AD_{15} and stores it temporarily, to be saved on the stack later in the acknowledge sequence. This word identifies the source of the interrupt or trap. For internal traps, the identifier is the first word of the trapped instruction. For external events, the identifier is the contents of the Data bus as sampled during T_3 of the acknowledge cycle. During nonvectored and nonmaskable interrupts, all 16 bits can represent peripheral device status information. For the vectored interrupt, the low byte is the jump vector, and the high byte can be used for extra status. For a $\overline{SAT}$ trap (assuming that a Zilog Z8010 Z-MMU Memory Management Unit is used) the high byte is the memory management unit identifier and the low byte is undefined.

After the acknowledge cycle, the $N/\overline{S}$ output indicates the automatic change to System mode.

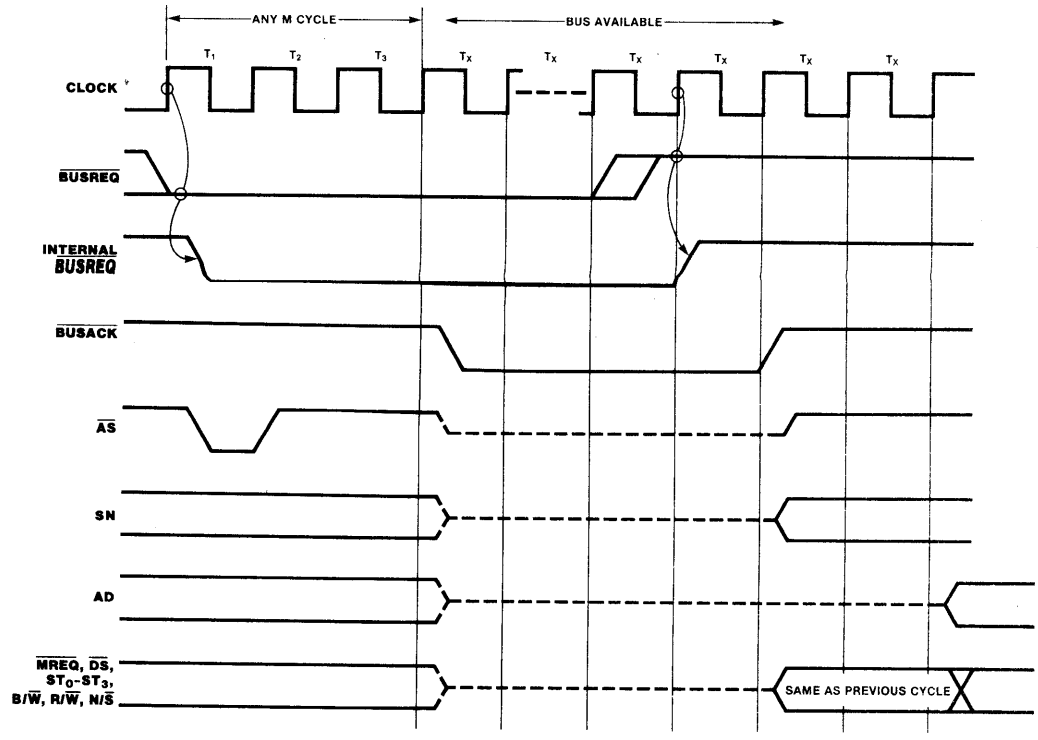


Figure 14. Bus Request/Acknowledge Timing

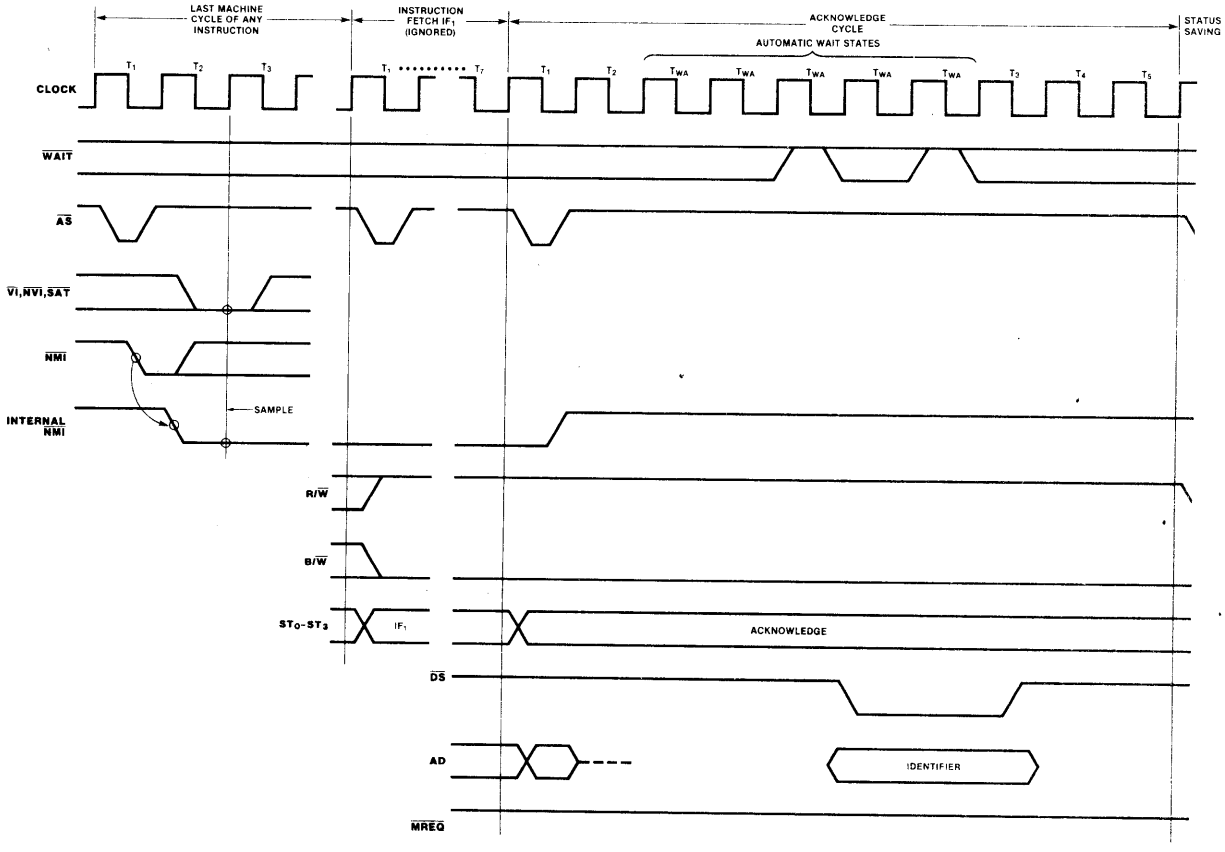


Figure 15. Interrupt and Segment/Address Translation Trap, Request/Acknowledge Timing

PIN DESCRIPTIONS

The Z8003 Z-VMPU is produced in a 48-pin package; the Z8004 Z-VMPU is produced in a 40-pin package. The pin functions of both the Z8003 and Z8004 are illustrated in Figure 16; the pin assignments are illustrated in Figure 17. The signal names assigned to the Z-VMPU I/O pins are listed alphabetically and are described in the following paragraphs.

$\overline{\text{ABORT}}$. *Abort Request (input, active Low).* This input is used to implement virtual memory. It is asserted by external circuitry when an address does not correspond to a location in main memory.

When $\overline{\text{ABORT}}$ is asserted with input $\overline{\text{SAT}}$ in the Z8003, or with input $\overline{\text{NMI}}$, $\overline{\text{VI}}$, or $\overline{\text{NVI}}$ in the Z8004, it initiates an Abort Interrupt in the Z-VMPU.

AD_0–AD_{15}. *Address/Data (inputs/outputs, active High, 3-state).* These multiplexed address and data lines are used both for I/O and memory.

$\overline{\text{AS}}$. *Address Strobe (output, active Low, 3-state).* The rising edge of $\overline{\text{AS}}$ indicates that addresses are valid.

$\overline{\text{BUSACK}}$. *Bus Acknowledge (output, active Low).* A Low on this line indicates that the Z-VMPU has relinquished control of the bus.

$\overline{\text{BUSREQ}}$. *Bus Request (input, active Low).* This line must be driven Low to request the bus from the Z-VMPU.

B/$\overline{\text{W}}$. *Byte/Word (output, Low* = Word, 3-state). This line defines the size of the data being transferred.

CLK. *System Clock (input).* CLK is a +5 V single-phase, time-base input.

$\overline{\text{DS}}$. *Data Strobe (output, active Low, 3-state).* This line strobes data in and out of the Z-VMPU.

$\overline{\text{MI}}$, $\overline{\text{MO}}$. *Multi-Micro In, Multi-Micro Out (input and output, active Low).* These two lines form a resource-request daisy chain that allows only one Z-VMPU in a multi-microprocessor system to access a shared resource at the same time.

$\overline{\text{MREQ}}$. *Memory Request (output, active Low, 3-state).* A Low on this line indicates that a memory reference is in progress.

$\overline{\text{NMI}}$. *Nonmaskable Interrupt (edge-triggered, input, active Low).* A High-to-Low transition on $\overline{\text{NMI}}$ requests a nonmaskable interrupt.

N/$\overline{\text{S}}$. *Normal/System Mode (output, Low* = System mode, 3-state). N/$\overline{\text{S}}$ indicates the current Z-VMPU operating mode (System or Normal).

$\overline{\text{NVI}}$. *Nonvectored Interrupt (input, active Low).* A Low on this line requests a nonvectored interrupt.

$\overline{\text{RESET}}$. *Reset (input, active Low).* A Low on this line resets the Z-VMPU.

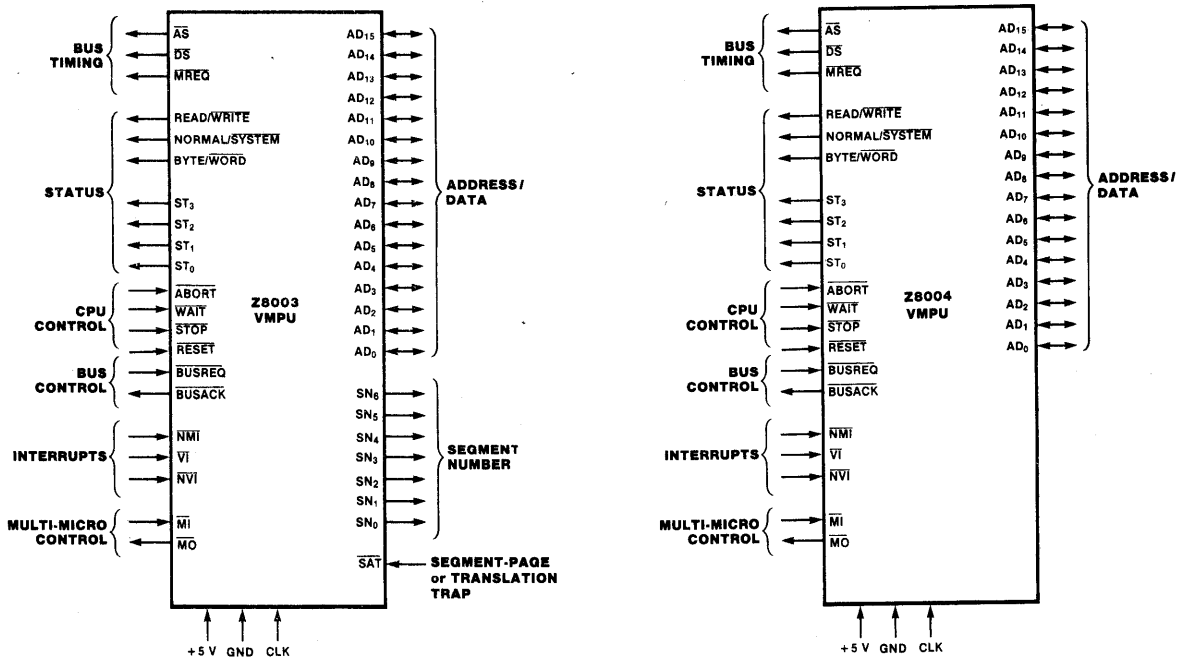


Figure 16. Pin Functions

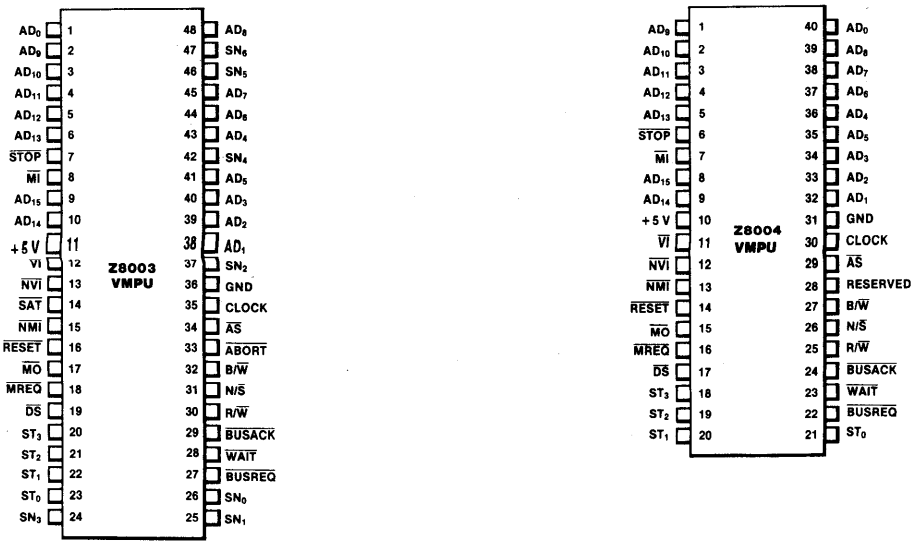


Figure 17. Pin Assignments

INSTRUCTION SET SUMMARY

The Z8003/04 instruction set is presented in the instruction set summary. This summary lists the mnemonics, operands, addressing modes, timing, and operation for each instruction.

Timing is given as the number of CPU clock cycles required for instruction execution. Timing requirements are given for the three possible addressing representations used in word, byte and long word operations:

- NS nonsegmented addresses
- SS segmented short-offset addresses
- SL segmented long-offset addresses

The SS and SL address representations apply only to those instructions for which the address of the operand is contained within the instruction itself. The only instructions of this type are those using the DA and X addressing modes.

With few exceptions, timing requirements are the same for all instructions in either segmented or nonsegmented mode, except for those instructions that employ the SS and SL addresses. The timing for these instructions will differ since the number of fetches needed to load the address, one word or two words, will vary.

NOTE

Timing values are given in the SS and SL columns of the instruction set summary for all addressing modes, even where the address representation does not apply. These values are given to indicate that the time requirements are the same for both segmented and nonsegmented modes.

INSTRUCTION SET SUMMARY

The Z8003/4 provides the following types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control

Load and Exchange

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
CLR **CLRB**	dst	R	7	7	7				**Clear** dst ← 0
		IR	8	8	8				
		DA	11	12	14				
		X	12	12	15				
EX **EXB**	R, src	R	6	6	6				**Exchange** R ↔ src
		IR	12	12	12				
		DA	15	16	18				
		X	16	16	19				
LD **LDB** **LDL**	R, src	R	3	3	3	5	5	5	**Load into Register** R ← src
		IM	7	7	7	11	11	11	
		IM	5(byte only)						
		IR	7	7	7	11	11	11	
		DA	9	10	12	12	13	15	
		X	10	10	13	13	13	16	
		BA	14	14	14	17	17	17	
		BX	14	14	14	17	17	17	
LD **LDB** **LDL**	dst, R	IR	8	8	8	11	11	11	**Load into Memory** (Store) dst ← R
		DA	11	12	14	14	15	17	
		X	12	12	15	15	15	18	
		BA	14	14	14	17	17	17	
		BX	14	14	14	17	17	17	
LD **LDB**	dst, IM	IR	11	11	11				**Load Immediate into Memory** dst ← IM
		DA	14	15	17				
		X	15	15	18				
LDA	R, src	DA	12	13	15				**Load Address** R ← source address
		X	13	13	16				
		BA	15	15	15				
		BX	15	15	15				
LDAR	R, src	RA	15	15	15				**Load Address Relative** R ← source address
LDK	R, src	IM	5	5	5				**Load Constant** R ← n (n = 0 ... 15)
LDM	R, src, n	IR	11	11	11				**Load Multiple** R ← src (n consecutive words) (n = 1 ... 16)
		DA	14	15	17	+3 n			
		X	15	15	18				
LDM	dst, R, n	IR	11	11	11				**Load Multiple** (Store Multiple) dst ← R (n consecutive words) (n = 1 ... 16)
		DA	14	15	17	+3 n			
		X	15	15	18				
LDR **LDRB** **LDRL**	R, src	RA	14	14	14	17	17	17	**Load Relative** R ← src (range -32768 ... +32767)
LDR **LDRB** **LDRL**	dst, R	RA	14	14	14	17	17	17	**Load Relative** (Store Relative) dst ← R (range -32768 ... +32767)
POP **POPL**	dst, IR	R	8	8	8	12	12	12	**Pop** dst ← IR Autoincrement contents of R
		IR	12	12	12	19	19	19	
		DA	16	16	18	23	23	25	
		X	16	16	19	23	23	26	
PUSH **PUSHL**	IR, src	R	9	9	9	12	12	12	**Push** Autodecrement contents of R IR ← src
		IM	12	12	12				
		IR	13	13	13	20	20	20	
		DA	13	14	16	21	21	23	
		X	14	14	17	21	21	24	

Arithmetic

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
ADC **ADCB**	R, src	R	5	5	5				**Add with Carry** R ← R + carry + src
ADD **ADDB** **ADDL**	R, src	R	4	4	4	8	8	8	**Add** R ← R + src
		IM	7	7	7	14	14	14	
		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP **CPB** **CPL**	R, src	R	4	4	4	8	8	8	**Compare with Register** R - src
		IM	7	7	7	14	14	14	
		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	
CP **CPB**	dst, IM	IR	11	11	11				**Compare with Immediate** dst - IM
		DA	14	15	17				
		X	15	15	18				
DAB	dst	R	5	5	5				**Decimal Adjust**
DEC **DECB**	dst, n	R	4	4	4				**Decrement by n** dst ← dst - n (n = 1 ... 16)
		IR	11	11	11				
		DA	13	14	16				
		X	14	14	17				
DIV **DIVL**	R, src	R	107	107	107	744	744	744	**Divide** (signed) Word: $R_{n+1} \leftarrow R_{n,n+1}$ + src R_n ← remainder Long Word: $R_{n+2,n+3} \leftarrow R_{n\ldots n+3}$ + src $R_{n,n+1}$ ← remainder
		IM	107	107	107	744	744	744	
		IR	107	107	107	744	744	744	
		DA	108	109	111	745	746	748	
		X	109	109	112	746	746	749	
EXTS **EXTSB** **EXTSL**	dst	R	11	11	11	11	11	11	**Extend Sign** Extend sign of low order half of dst through high order half of dst
INC **INCB**	dst, n	R	4	4	4				**Increment by n** dst ← dst + n (n = 1 ... 16)
		IR	11	11	11				
		DA	13	14	16				
		X	14	14	17				
MULT **MULTL**	R, src	R	70	70	70	282*	282*	282*	**Multiply** (signed) Word: $R_{n,n+1} \leftarrow R_{n+1}$ • src Long Word: $R_{n\ldots n+3} \leftarrow R_{n+2,n+3}$ • src * Plus seven cycles for each 1 in the absolute value of the low order word of the multiplicand
		IM	70	70	70	282*	282*	282*	
		IR	70	70	70	282*	282*	282*	
		DA	71	72	74	283*	284*	286*	
		X	72	72	75	284*	284*	287*	
NEG **NEGB**	dst	R	7	7	7				**Negate** dst ← -dst
		IR	12	12	12				
		DA	15	16	18				
		X	16	16	19				
SBC **SBCB**	R, src	R	5	5	5				**Subtract with Carry** R ← R - src - carry
SUB **SUBB** **SUBL**	R, src	R	4	4	4	8	8	8	**Subtract** R ← R - src
		IM	7	7	7	14	14	14	
		IR	7	7	7	14	14	14	
		DA	9	10	12	15	16	18	
		X	10	10	13	16	16	19	

Logical

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
AND **ANDB**	R, src	R	4	4	4				**And** R ← R AND src
		IM	7	7	7				
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				
COM **COMB**	dst	R	7	7	7				**Complement** dst ← NOT dst
		IR	12	12	12				
		DA	15	16	18				
		X	16	16	19				
OR **ORB**	R, src	R	4	4	4				**OR** R ← R OR src
		IM	7	7	7				
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				
TEST **TESTB** **TESTL**	dst	R	7	7	7	13	13	13	**Test** dst OR 0
		IR	8	8	8	13	13	13	
		DA	11	12	14	16	17	19	
		X	12	12	15	17	17	20	
TCC **TCCB**	cc, dst	R	5	5	5				**Test Condition Code** Set LSB if cc is true
XOR **XORB**	R, src	R	4	4	4				**Exclusive OR** R ← R XOR src
		IM	7	7	7				
		IR	7	7	7				
		DA	9	10	12				
		X	10	10	13				

Program Control

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
CALL	dst	IR	10	15	15				**Call Subroutine** Autodecrement SP @ SP ← PC PC ← dst
		DA	12	18	20				
		X	13	18	21				
CALR	dst	RA	10	15	15				**Call Relative** Autodecrement SP @ SP ← PC PC ← PC + dst (range −4094 to +4096)
DJNZ **DBJNZ**	R, dst	RA	11	11	11				**Decrement and Jump if Non-Zero** R ← R − 1 If R ≠ 0: PC ← PC + dst (range −254 to 0)
IRET*	–		13	16	16				**Interrupt Return** PS ← @ SP Autoincrement SP
JP	cc, dst	IR	10	15	15	(taken)			**Jump Conditional** If cc is true: PC ← dst
		IR	7	7	7	(not taken)			
		DA	7	8	10				
		X	8	8	11				

Program Control (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
RET	cc		10 7	13 7	13 7	(taken) (not taken)			**Return Conditional** If cc is true: PC ← @SP Autoincrement SP
SC	src	IM	33	39	39				**System Call** Autoincrement SP @ SP ← Old PS Push Instruction PS ← System Call PS
BIT **BITB**	dst,b	R IR DA X	4 8 10 11	4 8 11 11	4 8 13 14				**Test Bit Static** Z flag ← NOT dst bit specified by b
BIT **BITB**	dst,R	R	10	10	10				**Test Bit Dynamic** Z flag ← NOT dst bit specified by contents of R

*Privileged instructions. Executed in system mode only.

Bit Manipulation

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
RES **RESB**	dst,b	R IR DA X	4 11 13 14	4 11 14 14	4 11 16 17				**Reset Bit Static** Reset dst bit specified by b
RES **RESB**	dst,R	R	10	10	10				**Reset Bit Dynamic** Reset dst bit specified by contents R
SET **SETB**	dst,b	R IR	4 11	4 11	4 11				**Set Bit Static** Set dst bit specified by b
SET **SETB**	dst,R	R	10	10	10				**Set Bit Dynamic** Set dst bit specified by contents of R
TSET **TSETB**	dst	R IR DA X	7 11 14 15	7 11 15 15	7 11 17 18				**Test and Set** S flag ← MSB of dst dst ← all 1s

Rotate and Shift

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
RLDB	R,src	R	9	9	9				**Rotate Left Digit**
RRDB	R,src	R	9	9	9				**Rotate Right Digit**
RL **RLB**	dst,n	R R	6 for n = 1 7 for n = 2						**Rotate Left** Rotate dst by n bits (n = 1,2)
RLC **RLCB**	dst,n	R R	6 for n = 1 7 for n = 2						**Rotate Left through Carry** Rotate dst by n bits (n = 1,2)
RR **RRB**	dst,n	R R	6 for n = 1 7 for n = 2						**Rotate Right** Rotate dst by n bits (n = 1,2)

Rotate and Shift (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
RRC **RRCB**	dst, n	R R	6 for n = 1 7 for n = 2						**Rotate Right through Carry** Rotate dst by n bits (n = 1,2)
SDA **SDAB** **SDAL**	dst, R	R	(15 + 3n)			(15 + 3n)			**Shift Dynamic Arithmetic** *Shift dst left or right by contents of R*
SDL **SDLB** **SDLL**	dst, R	R	(15 + 3n)			(15 + 3n)			**Shift Dynamic Logical** Shift dst left or right by contents of R
SLA **SLAB** **SLAL**	dst, n	R	(13 + 3n)			(13 + 3n)			**Shift Left Arithmetic** Shift dst left by n bits
SLL **SLLB** **SLLL**	dst, n	R	(13 + 3n)			(13 + 3n)			**Shift Left Logical** Shift dst left by n bits
SRA **SRAB** **SRAL**	dst, n	R	(13 + 3n)			(13 + 3n)			**Shift Right Arithmetic** Shift dst right by n bits
SRL **SRLB** **SRLL**	dst, n	R	(13 + 3n)			(13 + 3n)			**Shift Right Logical** Shift dst right by n bits

Block Transfer and String Manipulation

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
CPD **CPDB**	R_X, src, R_Y, cc	IR	20	20	20				**Compare and Decrement** R_X – src Autodecrement src address $R_Y \leftarrow R_Y - 1$
CPDR **CPDRB**	R_X, src, R_Y, cc	IR	(11 + 9n)						**Compare, Decrement and Repeat** R_X – src Autodecrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPI **CPDRB**	R_X, src, R_Y, cc	IR	20	20	20				**Compare, Decrement and Repeat** R_X – src Autodecrement src address $R_Y \leftarrow R_Y - 1$
CPIR **CPIRB**	R_X, src, R_Y, cc	IR	(11 + 9n)						**Compare, Increment and Repeat** R_X – src Autoincrement src address $R_Y \leftarrow R_Y - 1$ Repeat until cc is true or $R_Y = 0$
CPSD **CPSDB**	dst, src, R, cc	IR	25	25	25				**Compare String and Decrement** dst ← src Autodecrement dst and src addresses R ← R – 1
CPSDR **CPSDRB**	dst, src, R, cc	IR	(11 + 14n)						**Compare String, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R – 1 Repeat until cc is true or R = 0

Block Transfer and String Manipulation (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
CPSI **CPSIB**	dst, src, R, cc	IR	25	25	25				**Compare String and Increment** dst – src Autoincrement dst and src addresses R ← R – 1
CPSIR **CPSIRB**	dst, src, R, cc	IR	(11 + 14n)						**Compare String, Increment and Repeat** dst – src Autoincrement dst and src addresses R ← R – 1 Repeat until cc is true or R = 0
LDD **LDDB**	dst, src, R	IR	20	20	20				**Load and Decrement** dst ← src Autodecrement dst and src addresses R ← R – 1
LDDR **LDDRB**	dst, src, R	IR	(11 + 9n)						**Load, Decrement and Repeat** dst ← src Autodecrement dst and src addresses R ← R – 1 Repeat until R = 0
LDI **LDIB**	dst, src, R	IR	20	20	20				**Load and Increment** dst ← src Autoincrement dst and src addresses R ← R – 1
LDIR **LDIRB**	dst, src, R	IR	(11 + 9n)						**Load, Increment and Repeat** dst ← src Autoincrement dst and src addresses R ← R – 1 Repeat until R = 0
TRDB	dst, src, R	IR	25	25	25				**Translate and Decrement** dst ← src (dst) Autodecrement dst address R ← R – 1
TRDRB	dst, src, R	IR	(11 + 14n)						**Translate, Decrement and Repeat** dst ← src (dst) Autodecrement dst address R ← R – 1 Repeat until R = 0
TRIB	dst, src, R	IR	25	25	25				**Translate and Increment** dst ← src (dst) Autoincrement dst address R ← R – 1
TRIRB	dst, src, R	IR	(11 + 14n)						**Translate, Increment and Repeat** dst ← src (dst) Autoincrement dst address R ← R – 1 Repeat until R = 0
TRTDB	src1, src2, R	IR	25	25	25				**Translate and Test, Decrement** RH1 ← src 2 (src1) Autodecrement src1 address R ← R – 1
TRTDRB	src1, src2, R	IR	(11 + 14n)						**Translate and Test, Decrement and Repeat** RH1 ← src2 (src1) Autodecrement src1 address R ← R – 1 Repeat until R = 0 or RH1 = 0

Block Transfer and String Manipulation (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
TRTIB	src1,src2,R	IR	25	25	25				**Translate and Test, Increment** RH1 ← src2 (src1) Autoincrement src1 address R ← R - 1
TRTIRB	src1,src2,R	IR	(11 + 14n)						**Translate and Test, Increment and Repeat** RH1 ← src2 (src1) Autoincrement src1 address R ← R - 1 Repeat until R = 0 or RH1 = 0

Input/Output

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
IN* **INB***	R,src	IR DA	10 12	10 12	10 12				**Input** R ← src
IND* **INDB***	dst,src,R	IR	21	21	21				**Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
INDR* **INDRB***	dst,src,R	IR	(11 + 10n)						**Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
INI* **INIB***	dst,src,R	IR	21	21	21				**Input and Increment** dst ← src Autoincrement dst address R ← R - 1
INIR* **INIRB***	dst,src,R	IR	(11 + 10n)						**Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
OUT* **OUTB***	dst,R	IR DA	10 12	10 12	10 12				**Output** dst ← R
OUTD* **OUTDB***	dst,src,R	IR	21	21	21				**Output and Decrement** dst ← src Autodecrement src address R ← R - 1
OTDR* **OTDRB***	dst,src,R	IR	(11 + 10n)						**Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
OUTI* **OUTIB***	dst,src,R	IR	21	21	21				**Output and Increment** dst ← src Autoincrement src address R ← R - 1

Input/Output (Continued)

Mnemonics	Operands	Addr. Modes	Clock Cycles: Word, Byte NS	Word, Byte SS	Word, Byte SL	Long Word NS	Long Word SS	Long Word SL	Operation
OTIR* **OTIRB***	dst, src, R	IR	(11 + 10n)						**Output, Increment and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0
SIN* **SINB***	R, src	DA	12	12	12				**Special Input** R ← src
SIND* **SINB***	dst, src, R	IR	21	21	21				**Special Input and Decrement** dst ← src Autodecrement dst address R ← R - 1
SINDR* **SINDRB***	dst, src, R	IR	(11 + 10n)						**Special Input, Decrement and Repeat** dst ← src Autodecrement dst address R ← R - 1 Repeat until R = 0
SINI* **SINIB***	dst, src, R	IR	21	21	21				**Special Input and Increment** dst ← src Autoincrement dst address R ← R - 1
SINIR* **SINIRB***	dst, src, R	IR	(11 + 10n)						**Special Input, Increment and Repeat** dst ← src Autoincrement dst address R ← R - 1 Repeat until R = 0
SOUT* **SOUTB***	dst, src	DA	12	12	12				**Special Output** dst ← src
SOUTD* **SOUTDB***	dst, src, R	IR	21	21	21				**Special Output and Decrement** dst ← src Autodecrement src address R ← R - 1
SOTDR* **SOTDRB***	dst, src, R	IR	(11 + 10n)						**Special Output, Decrement and Repeat** dst ← src Autodecrement src address R ← R - 1 Repeat until R = 0
SOUTI* **SOUTIB***	dst, src, R	IR	21	21	21				**Special Output and Increment** dst ← src Autoincrement src address R ← R - 1
SOTIR* **SOTIRB***	dst, src, R	R	(11 + 10n)						**Special Output, Increment and Repeat** dst ← src Autoincrement src address R ← R - 1 Repeat until R = 0

*Privileged instructions. Executed in system mode only.

CPU Control

Mnemonics	Operands	Addr. Modes	Clock Cycles Word, Byte NS	SS	SL	Long Word NS	SS	SL	Operation
COMFLG	flags		7	7	7				**Complement Flag** (Any combination of C,Z,S,P/V)
DI*	int		7	7	7				**Disable Interrupt** (Any combination of NVI, VI)
EI*	int		7	7	7				**Enable Interrupt** (Any combination of NVI, VI)
HALT*			(8 + 3n)						**HALT**
LDCTL*	CTLR,src	R	7	7	7				**Load into Control Register** CTLR ← src
LDCTL*	dst,CTLR	R	7	7	7				**Load from Control Register** dst ← CTLR
LDCTLB	FLGR,src	R	7	7	7				**Load into Flag Byte Register** FLGR ← src
LDCTLB	dstFLGR	R	7	7	7				**Load from Flag Byte Register** dst ← FLGR
LDPS*	src	IR DA X	12 16 17	16 20 20	16 22 23				**Load Program Status** PS ← src
MBIT*			7	7	7				**Test Multi-Micro Bit** Set S if MI is Low; clear S if MI is High
MREQ*	dst	R	(12 + 7n)						**Multi-Micro Request**
MRES*			5	5	5				**Multi-Micro Reset**
MSET*			5	5	5				**Multi-Micro Set**
NOP			7	7	7				**No Operation**
RESFLG	flag		7	7	7				**Reset Flag** (Any combination of C,Z,S,P/V)
SETFLG	flag		7	7	7				**Set Flag** (Any combination of C,Z,S,P/V)

*Privileged instructions. Executed in system mode only.

Extended Instructions

Function	Addr. Modes	Clock Cycles NS	SS	SL	Operation
Memory ← EPU	IR	(11 + 3n)	(11 + 3n)	(11 + 3n)	**Load Memory from EPU**
	X	(15 + 3n)	(15 + 3n)	(18 + 3n)	Write n words from EPU into memory
	DA	(14 + 3n)	(15 + 3n)	(17 + 3n)	
EPU ← Memory	IR	(11 + 3n)	(11 + 3n)	(11 + 3n)	**Load EPU from Memory**
	X	(15 + 3n)	(15 + 3n)	(18 + 3n)	Read n words from memory into EPU
	DA	(14 + 3n)	(15 + 3n)	(17 + 3n)	
CPU ← EPU Registers		(11 + 4n)	(11 + 4n)	(11 + 4n)	**Load VMPU from EPU** Transfer n words from EPU to Z-VMPU registers
EPU ← CPU Registers		(11 + 4n)	(11 + 4n)	(11 + 4n)	**Load EPU from VMPU** Transfer n words from Z-VMPU registers to EPU
Flags ← EPU		15	15	15	**Load FCW from EPU** Load information from EPU into flags of the Z-VMPU's Flag and Control Word
EPU ← Flags		15	15	15	**Load EPU from FCW** Transfer information from Z-VMPU's Flag and Control Word to EPU
EPU Internal Operations		(11 + 4n)	(11 + 4n)	(11 + 4n)	**Internal EPU Operations** Z-VMPU treats this template as a "no-operations"; it is typically used to initiate an internal EPU operation. The character is a field in the instruction.

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs with respect to GND . . . . . −0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition beyond those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

Standard test temperature/operating voltage ranges are presented below. All voltages are referenced to GND. Positive current flows into the referenced pin.

- 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- −40°C to +85°C, +4.75 V ≤ V_{CC} ≤ +5.25V
- −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

All ac parameters assume a load capacitance of 100 pF max, except for parameter 6, which has a load capacitance of 50 pF max. Timing references between two output signals assume a load difference of 50 pF max.

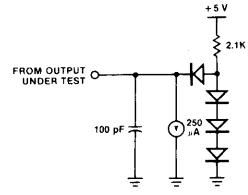


Figure 18. Standard Test Load

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}−0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 A
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	A	0.4 V_{IN} +2.4 V
I_{OL}	Output Leakage		±10	A	0.4 V_{OUT} +2.4 V
I_{CC}	V_{CC} Supply Current		300	mA	

AC CHARACTERISTICS†

			Z8003/Z8004 (4 MHz)		Z8003A/Z8004A (6 MHz)		Z8003B/Z8004B (10 MHz)	
Number	Symbol	Parameter	Min	Max	Min	Max	Min	Max
1	TcC	Clock Cycle Time	250	2000	165	2000	100	2000
2	TwCh	Clock Width (High)	105	2000	70	2000	40	
3	TwCl	Clock Width (Low)	105	2000	70	2000	40	
4	TfC	Clock Fall Time		20		10		10
5	TrC	Clock Rise Time		20		15		10
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130		110		70
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		10		5	
8	TdC(Bz)	Clock ↑ to Bus Float		65		55		40
9	TdC(A)	Clock ↑ to Address Valid		100		75		50
10	TdC(Az)	Clock ↑ to Address Float		65		55		40
11	TdA(DR)	Address Valid to Read Data Required Valid		475*		305*		180*
12	TsDR(C)	Read Data to Clock ↓ Setup Time	30		20		10	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80*		45*		20*	
14	TdC(DW)	Clock ↑ to Write Data Valid		100		75		50
15	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		0	
16	TdDW(DS)	Write Data Valid to $\overline{DS}$ ↑ Delay	295*		195*		110*	

2084-018

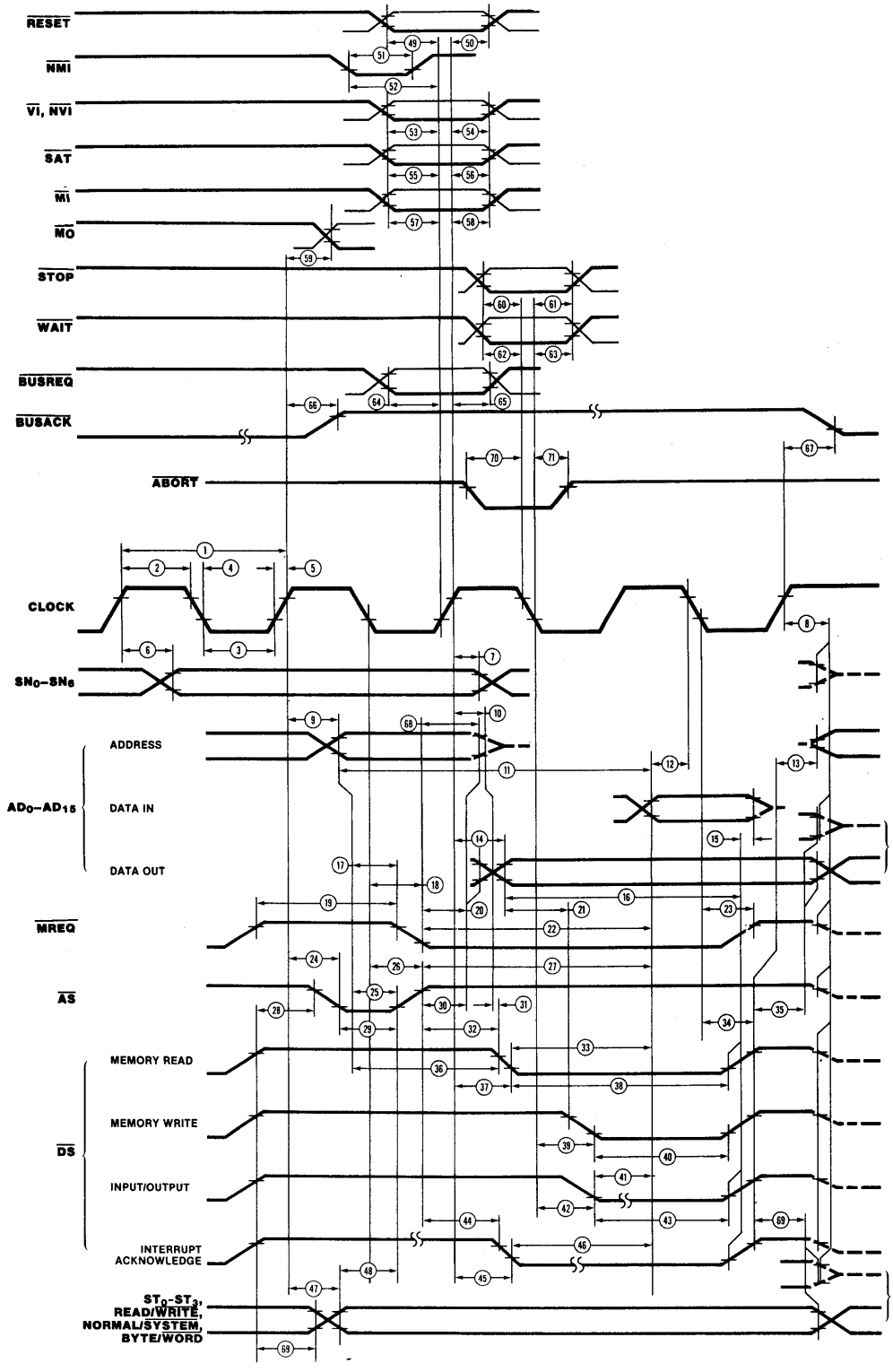
AC CHARACTERISTICS† (Continued)

Number	Symbol	Parameter	Z8003/Z8004 (4 MHz) Min	Z8003/Z8004 (4 MHz) Max	Z8003A/Z8004A (6 MHz) Min	Z8003A/Z8004A (6 MHz) Max	Z8003B/Z8004B (10 MHz) Min	Z8003B/Z8004B (10 MHz) Max
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55		35*		20*	
18	TdC(MR)	Clock ↓ to $\overline{MREQ}$ ↓ Delay		80		70		40
19	TwMRh	$\overline{MREQ}$ Width (High)	210*		135*		80*	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70*		35*		20*	
21	TdDW(DSW)	Write Data Valid to $\overline{DS}$ ↓ (Write) Delay	55*		35*		15*	
22	TdMR(DR)	$\overline{MREQ}$ ↓ to Read Data Required Valid	375*		230*		140*	
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		80		60		45
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		60		40
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55*		35*		20*	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		80		40
27	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid	360*		220*		140*	
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70*		35*		15*	
29	TwAS	$\overline{AS}$ Width (Low)	85*		55*		30*	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	70*		45*		20*	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0		0	
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	80*		55*		30*	
33	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid	205*		130*		70*	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65		45
35	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid	75*		45*		25*	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180*		110*		65*	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		85		60
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275*		185*		110*	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		80		60
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185*		110*		75*	
41	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid	330*		210*		120*	
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		90		60
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410*		255*		160*	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065*		690*		410*	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		85		65
46	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Required Delay	455*		295*		165*	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		85		60
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	50*		30*		10*	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		70		50	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		0		0	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		70		50	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		70		50	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		50		40	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	20		20		10	
55	TsSGT(C)	$\overline{SAT}$ to Clock ↑ Setup Time	70		55		40	
56	ThSGT(C)	$\overline{SAT}$ to Clock ↑ Hold Time	0		0		0	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		110		80	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		0		0	
59	TdC(MO)	Clock ↑ to $\overline{MO}$ Delay		120		85		70
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		80		50	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		0		0	
62	TsW(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		30		20	
63	ThW(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		10		5	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	90		80		60	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		10		5	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		100		75		60
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		100		75		60
68	TwA	Address Valid Width	150*		95*		50*	
69	TdDS(S)	$\overline{DS}$ ↑ to STATUS Not Valid	80*		55*		30*	
70	TsABT(C)	$\overline{ABORT}$ ↓ to Clock ↑ Setup Time	50		30		25	
71	ThABT(C)	$\overline{ABORT}$ ↓ to Clock ↓ Hold Time	0		0		0	

*Clock-cycle-time-dependent characteristics. See table on following page.

†Timings are preliminary and subject to change. Units in nanoseconds(ns).

COMPOSITE AC TIMING DIAGRAM



CLOCK-CYCLE-TIME-DEPENDENT CHARACTERISTICS

Number	Symbol	Z8003 Equation	Z8003A Equation	Z8003B Equation
11	TdA(DR)	2TcC + TwCh – 130 ns	2TcC + TwCh – 95 ns	2TcC + TwCh – 60 ns
13	TdDS(A)	TwCl – 25 ns	TwCl – 25 ns	TwCl – 20 ns
16	TdDW(DS)	TcC + TwCh – 60 ns	TcC + TwCh – 40 ns	TcC + TwCh – 30 ns
17	TdA(MR)	TwCh – 50 ns	TwCh – 35 ns	TwCh – 20 ns
19	TwMRh	TcC – 40 ns	TcC – 30 ns	TcC – 20 ns
20	TdMR(A)	TwCl – 35 ns	TwCl – 35 ns	TwCl – 20 ns
21	TdDW(DSW)	TwCh – 50 ns	TwCh – 35 ns	TwCh – 25 ns
22	TdMR(DR)	2TcC – 130 ns	2TcC – 100 ns	2TcC – 60 ns
25	TdA(AS)	TwCh – 50 ns	TwCh – 35 ns	TwCh – 20 ns
27	TdAS(DR)	2TcC – 140 ns	2TcC – 110 ns	2TcC – 60 ns
28	TdDS(AS)	TwCl – 35 ns	TwCl – 35 ns	TwCl – 25 ns
29	TwAS	TwCh – 20 ns	TwCh – 15 ns	TwCh – 10 ns
30	TdAS(A)	TwCl – 35 ns	TwCl – 25 ns	TwCl – 20 ns
32	TdAS(DSR)	TwCl – 25 ns	TwCl – 15 ns	TwCl – 10 ns
33	TdDSR(DR)	TcC + TwCh – 150 ns	TcC + TwCh – 105 ns	TcC + TwCh – 70 ns
35	TdDS(DW)	TwCl – 30 ns	TwCl – 25 ns	TwCl – 15 ns
36	TdA(DSR)	TcC – 70 ns	TcC – 55 ns	TcC – 35 ns
38	TwDSR	TcC + TwCh – 80 ns	TcC + TwCh – 50 ns	TcC + TwCh – 30 ns
40	TwDSW	TcC – 65 ns	TcC – 55 ns	TcC – 25 ns
41	TdDSI(DR)	2TcC – 170 ns	2TcC – 120 ns	2TcC – 80 ns
43	TwDS	2TcC – 90 ns	2TcC – 75 ns	2TcC – 40 ns
44	TdAS(DSA)	4TcC + TwCl – 40 ns	4TcC + TwCl – 40 ns	4TcC + TwCl – 30 ns
46	TdDSA(DR)	2TcC + TwCh – 150 ns	2TcC + TwCh – 105 ns	2TcC + TwCh – 75 ns
48	TdS(AS)	TwCh – 55 ns	TwCh – 40 ns	TwCh – 30 ns
68	TwA	TcC – 90 ns	TcC – 70 ns	TcC – 50 ns
69	TdDS(S)	TwCl – 25 ns	TwCl – 15 ns	TwCl – 10 ns

ORDERING INFORMATION

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8003	CS	4.0 MHz	Z-VMPU (48-pin)	Z8004	CS	4.0 MHz	Z-VMPU (40-pin)
Z8003A	CS	6.0 MHz	(same as above)	Z8004A	CS	6.0 MHz	(same as above)
Z8003B	CS	10.0 MHz	(same as above)	Z8004B	CS	10.0 MHz	(same as above)

NOTES: C = Ceramic; S = 0°C to 70°C

Z8010
Z8000™ Z-MMU Memory Management Unit

Zilog

Product Specification

September 1983

Features

- Dynamic segment relocation makes software addresses independent of physical memory addresses.
- Sophisticated memory-management features include access validation that protects memory areas from unauthorized or unintentional access, and a write-warning indicator that predicts stack overflow.
- For use with both Z8001 and Z8003 CPU.
- 64 variable-sized segments from 256 to 65,536 bytes can be mapped into a total physical address space of 16M bytes; all 64 segments are randomly accessible.
- Multiple MMUs can support several translation tables for each Z8001/3 address space.
- MMU architecture supports multi-programming systems and virtual memory implementations.

General Description

The Z8010 Memory Management Unit (MMU) manages the large 8M byte addressing spaces of the Z8001 CPU. The MMU provides dynamic segment relocation as well as numerous memory protection features.

Dynamic segment relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is actually located in the physical memory. It also provides a flexible, efficient method for supporting multi-programming systems. The MMU uses a translation table to transform the 23-bit logical address output from the Z8001 CPU into a 24-bit address for the physical memory. (Only logical memory addresses go to an MMU for translation; I/O addresses and data, in general, must by pass this component.)

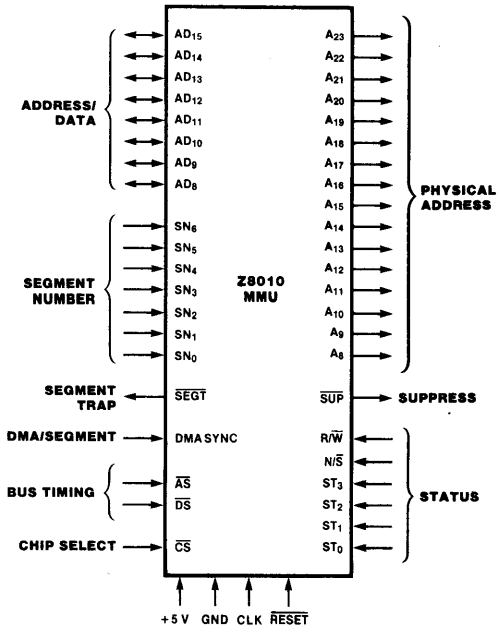


Figure 1. Pin Functions

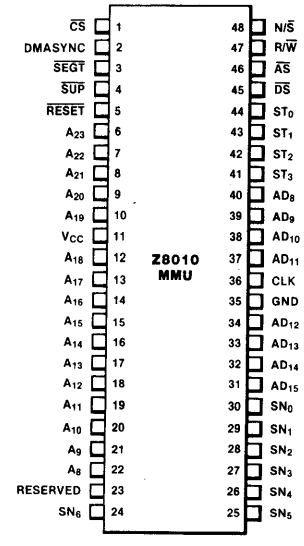


Figure 2. Pin Assignments

General Description (Continued)

Memory segments are variable in size from 256 bytes to 64K bytes, in increments of 256 bytes. Pairs of MMUs support the 128 segment numbers available for the various Z8001 CPU address spaces. Within an address space, any number of MMUs can be used to accommodate multiple translation tables for System and Normal operating modes, or to support more sophisticated memory-management systems.

MMU memory-protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each segment. A segment is assigned a number of attributes when its descriptor is entered into the MMU. When a memory reference is made, these attributes are checked against the status information supplied by the Z8001/3 CPU. If a mismatch occurs, a trap is generated and the CPU is interrupted. The CPU can then check the status registers of the MMU to determine the cause.

Segments are protected by modes of permitted use, such as read only, system only, execute only and CPU-access only. Other segment management features include a write-warning zone useful for stack operations and status flags that record read or write accesses to each segment.

The MMU is controlled via 22 Special I/O instructions from the Z8000 CPU in System mode. With these instructions, system software can assign program segments to arbitrary memory locations, restrict the use of segments and monitor whether segments have been read or written.

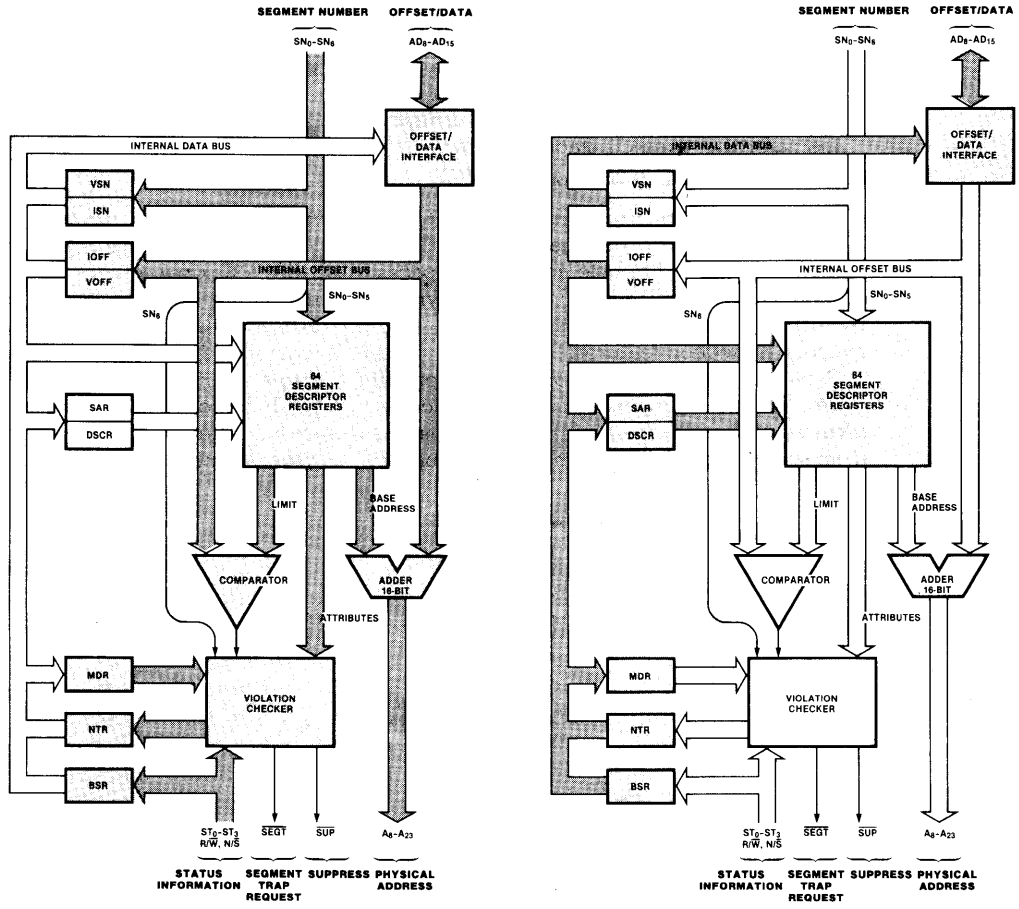


Figure 3. The shaded areas in these block diagrams illustrate the resources used in the two modes of MMU operation. In the Address Translation Mode shown on the left, addresses are translated automatically. In the Command Mode shown on the right, specific registers are accessed using Special I/O commands.

Segmented Addressing

A segmented addressing space—compared with linear addressing—is closer to the way a programmer uses memory because each procedure and data set can reside in its own segment.

The 8M byte Z8001 addressing spaces are divided into 128 relocatable segments of up to 64K bytes each. A 23-bit segmented address uses a 7-bit segment address to point to the segment, and a 16-bit offset to address any byte relative to the beginning of the segment. The two parts of the segmented address may be manipulated separately.

The MMU divides the physical memory into 256-byte blocks. Segments consist of physically contiguous blocks. Certain segments may be designated so that writes into the last block generate a warning trap. If such a segment is used as a stack, this warning can be used to increase the segment size and prevent a stack overflow error.

The addresses manipulated by the programmer, used by instructions and output by the Z8001 are called *logical addresses*. The MMU takes the logical addresses and transforms them into the *physical addresses* required for accessing the memory (Figure 4). This address transformation process is called *relocation*.

The relocation process is transparent to user software. A translation table in the MMU associates the 7-bit segment number with the base address of the physical memory segment. The 16-bit logical address offset is added to the physical base address to obtain the actual physical memory location. Because a base address always has a low byte equal to zero, only the high-order 16 bits are stored in the MMU and used in the addition. Thus the low-order byte of the physical memory location is the same as the low-order byte of the logical address offset. This low-order byte therefore bypasses the MMU, thus reducing the number of pins required.

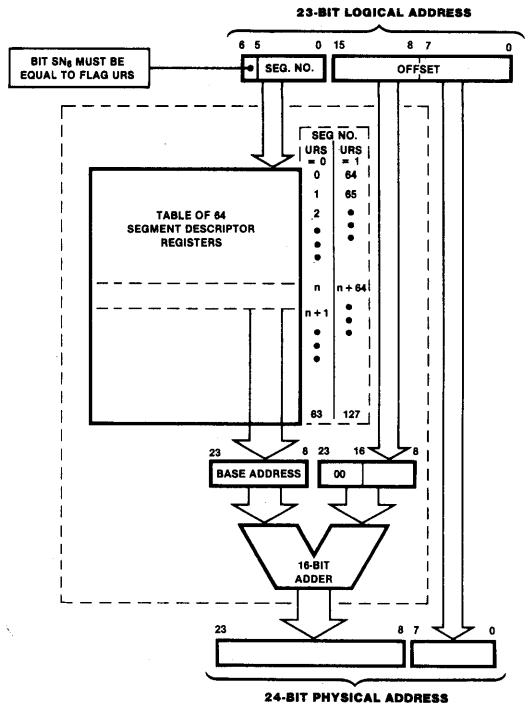


Figure 4. Logical-to-Physical Address Translation

Memory Protection

Each memory segment is assigned several attributes that are used to provide memory access protection. A memory request from the Z8001/3 CPU is accompanied by status information that indicates the attributes of the memory request. The MMU compares the memory request attributes with the segment attributes and generates a Trap Request whenever it detects an attribute violation. Trap Request informs the Z8001/3 CPU and the system control program of the violation so that appropriate action can be taken to recover. The MMU also generates the Suppress signal $\overline{SUP}$ in the event of an access violation. Suppress can be used by a memory system to inhibit stores into the memory and thus protect the contents of the memory from erroneous changes.

Five attributes can be associated with each segment. When an attempted access violates any one of the attributes associated with a segment, a Trap Request and a Suppress signal are generated by the MMU. These attributes are read only, execute only, system access only, inhibit CPU accesses and inhibit DMA accesses.

Segments are specified by a base address and a range of legal offsets to this base address. On each access to a segment, the offset is checked against this range to insure that the access falls within the allowed range. If an access that lies outside the segment is attempted, Trap Request and Suppress are generated.

Normally the legal range of offsets within a segment is from 0 to 256N + 255 bytes, where $0 \leq N \leq 255$. However, a segment may be specified so that legal offsets range from 256N to 65,535 bytes, where $0 \leq N \leq 255$. The later type of segment is useful for stacks since the Z8000 stack manipulation instructions cause stacks to grow toward lower memory locations. Thus when a stack grows to the limit of its allocated segment, additional memory can be allocated on the correct end of the segment. As an aid in maintaining stacks, the MMU detects when a write is performed to the lowest allocated 256 bytes of these segments and generates a Trap Request. No Suppress signal is generated so the write is allowed to proceed. This write warning can then be used to indicate that more memory should be allocated to the segment.

MMU Register Organization

The MMU contains three types of registers: Segment Descriptor, Control and Status. A set of 64 Segment Descriptor Registers supplies the information needed to map logical memory addresses to physical memory locations. The segment number of a logical address determines which Segment Descriptor Register is used in address translation. Each Descriptor Register also contains the necessary information for checking that the segment location referenced is within the bounds of the segment and that the type of reference is permitted. It also indicates whether the segment has been read or written.

In addition to the Segment Descriptor Registers, the Z8010 MMU contains three 8-bit control registers for programming the device and six 8-bit status registers that record information in the event of an access violation.

Segment Descriptor Registers. Each of the 64 Descriptor Registers contains a 16-bit base address field, an 8-bit limit field and an 8-bit attribute field (Figure 5). The base address field is subdivided into high- and low-order bytes that are loaded one byte at a time when the descriptor is initialized. The limit field contains a value N that indicates N + 1 blocks of 256 bytes have been allocated to the segment.*

The attribute field contains eight flags (Figure 6). Five are related to protecting the segment against certain types of access, one indicates the special structure of the segment, and two encode the types of accesses that have been made to the segment. A flag is set when its value is 1. The following brief descriptions indicate how these flags are used.

Read-Only (RD). When this flag is set, the segment is read only and is protected against any write access.

System-Only (SYS). When this flag is set, the segment can be accessed only in System mode, and is protected against any access in Normal mode.

CPU-Inhibit (CPUI). When this flag is set, the segment is not accessible to the currently executing process, and is protected against any memory access by the CPU. The segment is, however, accessable under DMA.

Execute-Only (EXC). When this flag is set, the segment can be accessed only during an instruction fetch or access by the relative addressing mode cycle, and thus is protected against any access during other cycles.

DMA-Inhibit (DMAI). When this flag is set, the segment can be accessed only by the CPU, and thus is protected against any access under DMA.

Direction and Warning (DIRW). When this flag is set, the segment memory locations are considered to be organized in descending order and each write to the segment is checked for access to the last 256-byte block. Such an access generates a trap to warn of potential segment overflow, but no Suppress signal is generated.

Changed (CHG). When this flag is set, the segment has been changed (written). This bit is set automatically during any write access to this segment if the write access does not cause any violation.

Referenced (REF). When this flag is set, the segment has been referenced (either read or written). This bit is set automatically during any access to the segment if the access does not cause a violation.

*In the stack mode, segment size is 64K-256N.

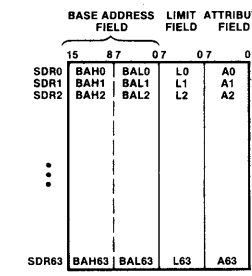


Figure 5. Segment Descriptor Registers

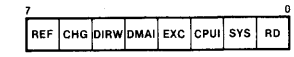


Figure 6. Attribute Field in Segment Descriptor Register

Control Registers. The three user-accessible 8-bit control registers in the MMU direct the functioning of the MMU (Figure 7). The Mode Register provides a sophisticated method for selectively enabling MMUs in multiple-MMU configurations. The Segment Address Register (SAR) selects a particular Segment Descriptor Register to be accessed during a control operation. The Descriptor Selection Counter Register points to a byte within the Segment Descriptor Register to be accessed during a control operation.

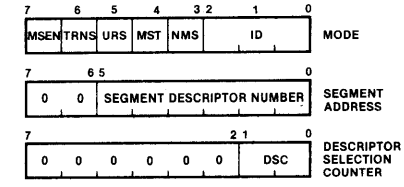


Figure 7. Control Registers

The Mode Register contains a 3-bit identification field (ID) that distinguishes among eight enabled MMUs in a multiple-MMU configuration. This field is used during the segment trap acknowledge sequence (refer to the section on Segment Trap and Acknowledge). In addition, the Mode Register contains five flags.

Multiple Segment Table (MST). This flag indicates whether multiple segment tables are present in the hardware configuration. When this flag is set, more than one table is present and the N/$\overline{S}$ line must be used to determine whether the MMU contains the appropriate table.

Normal Mode Select (NMS). This flag indicates whether the MMU is to translate addresses when the N/$\overline{S}$ line is High or Low. If the MST flag is set, the N/$\overline{S}$ line must match the NMS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

MMU Register Organization (Continued)

Upper Range Select (URS). This flag is used to indicate whether the MMU contains the lower-numbered segment descriptors or the higher-numbered segment descriptors. The most significant bit of the segment number must match the URS flag for the MMU to translate segment addresses, otherwise the MMU Address lines remain 3-stated.

Translate (TRNS). This flag indicates whether the MMU is to translate logical program addresses to physical memory locations or is to pass the logical addresses unchanged to the memory and without protection checking. In the non-translation mode, the most significant byte of the output is the 7-bit segment number and the most significant bit is 0. When this flag is set, the MMU performs address translation and attribute checking.

Master Enable (MSEN). This flag enables or disables the MMU from performing its address translation and memory protection functions. When this flag is set, the MMU performs these tasks; when the flag is clear the Address lines of the MMU remain 3-stated.

The Segment Address Register (SAR) points to one of the 64 segment descriptors. Control commands to the MMU that access segment descriptors implicitly use this pointer to select one of the descriptors. This register has an auto-incrementing capability so that multiple descriptors can be accessed in a block read/write fashion.

The Descriptor Selection Counter Register holds a 2-bit counter that indicates which byte in the descriptor is being accessed during the reading or writing operation. A value of zero in this counter indicates the high-order byte of the base address field is to be accessed, one indicates the low-order byte of the base address, two indicates the limit field and three indicates the attribute field.

Status Registers. Six 8-bit registers contain information useful in recovering from memory access violations (Figure 8). The Violation Type Register describes the conditions that generated the trap. The Violation Segment Number and Violation Offset Registers record the most-significant 15 bits of the logical address that causes a trap. The Instruction Segment Number and Offset Registers record the most-significant 15 bits of the logical address of the last instruction fetched before the first accessing violation. These two registers can be used in conjunction with external circuitry that records the low-order offset byte. At the time of the addressing violation, the Bus Cycle Status Register records the bus cycle status (status code, read/write mode and normal/system mode).

The MMU generates a Trap Request for two general reasons: either it detects an access violation, such as an attempt to write into a read-only segment, or it detects a warning condition, which is a write into the lowest 256 bytes of a segment with the DIRW flag set. When a violation or warning condition is detected, the MMU generates a Trap Request and automatically sets the appropriate flags. The eight flags in the Violation Type Register describe the cause of a trap.

Read-Only Violation (RDV). Set when the CPU attempts to access a read-only segment and the R/$\overline{W}$ line is Low.

System Violation (SYSV). Set when the CPU accesses a system-only segment and the $\overline{N}$/S line is High.

CPU-Inhibit Violation (CPUIV). Set when the CPU attempts to access a segment with the CPU-inhibit flag set.

Execute-Only Violation (EXCV). Set when the CPU attempts to access an execute-only segment in other than an instruction fetch or load relative instructions cycle.

Segment Length Violation (SLV). Set when an offset falls outside of the legal range of a segment.

Primary Write Warning (PWW). Set when an access is made to the lowest 256 bytes of a segment with the DIRW flag set.

Secondary Write Warning (SWW). Set when the CPU pushes data into the last 256 bytes of the system stack and EXCV, CPUIV, SLV, SYSV, RDV or PWW is set. Once this flag is set, subsequent write warnings for accessing the system stack do not generate a Segment Trap request.

Fatal Condition (FATL). Set when any other flag in the Violation Type Register is set and either a violation is detected or a write warning condition occurs in Normal mode. This flag is not set during a stack push in System mode that results in a warning condition. This flag indicates a memory access error has occurred in the trap processing routine. Once set, no Trap Request signals are generated on subsequent violations. However, Suppress signals are generated on this and subsequent CPU violations until the FATL flag has been reset.

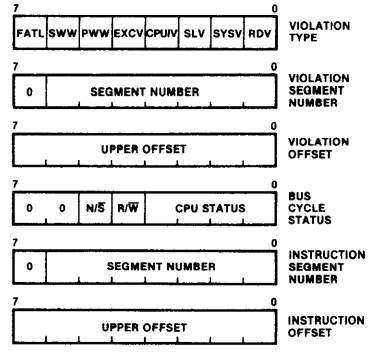


Figure 8. Status Registers

Segment Trap and Acknowledge

The Z8010 MMU generates a Segment Trap when it detects an access violation or a write warning condition. In the case of an access violation, the MMU also activates Suppress, which can be used to inhibit memory writes and to flag special data to be returned on a read access. Segment Trap remains Low until a Trap Acknowledge signal is received. If a CPU-generated violation occurs, Suppress is asserted for that cycle and all subsequent CPU instruction execution cycles until the end of the instruction. Intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only—no Segment Trap Requests are ever generated during DMA cycles.

Segment traps to the Z8001/3 CPU are handled similarly to other types of interrupts. To service a segment trap, the CPU issues a segment trap acknowledge cycle. The acknowledge cycle is always preceded by an instruction fetch cycle that is ignored (the MMU has been designed so that this dummy cycle is ignored). During the acknowledge cycle all enabled MMUs use the Address/Data lines to indicate their status. An MMU that has generated a Segment Trap Request outputs a 1 on the A/D line associated with the number in its ID field; an MMU that has not generated a segment trap request outputs a 0 on its associated A/D line. A/D lines for which no MMU is associated remain 3-stated. During a segment trap acknowledge cycle, an MMU uses A/D line 8 + i if its ID field is i.

Following the acknowledge cycle the CPU automatically pushes the Program Status onto the system stack and loads another Program Status from the Program Status Area. The Segment Trap line is reset during the segment trap acknowledge cycle. Suppress is not generated during the stack push. If the store creates a write warning condition, a Segment Trap Request is generated and is serviced at the end of the Program Status swap. The SWW flag is also set. Servicing this second Segment Trap Request also creates a write warning condition, but because the SWW flag is set, no Segment Trap Request is generated. If a violation rather than a write warning occurs during the Program Status swap, the FATL flag is set rather than the SWW flag. Subsequent violations cause Suppress to be asserted but not Segment Trap Request. Without the SWW and FATL flags, trap processing routines that generate memory violations would repeatedly be interrupted and called to process the trap they created.

The CPU routine to process a trap request should first check the FATL flag to determine if a fatal system error has occurred. If not, the SWW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the Violation Type Register reset.

Virtual Memory

Several features of the MMU can be used in conjunction with external circuitry to support virtual memory for the Z8001/3. Segment Trap Request can be used to signal the CPU in the event that a segment is not in primary memory. The CPU-Inhibit Flag can be used to indicate whether a segment is in the memory or in secondary storage. The Changed and Altered Flags in the attribute field for each segment can aid in implementing efficient segment management policies. The Status Registers can be used in recovering from virtual memory access faults.

Multiple MMUs

MMU architecture directly supports two methods for multiple MMU configurations. The first approach extends single-MMU capability for handling 64 segments to a dual-MMU configuration that manages the 128 different segments the Z8001/3 can address. This scheme uses the URS flag in the Mode Register in connection with the high-order bit of the segment number (SN_6).

The second approach uses several MMUs to implement multiple translation tables. Multiple tables can be used to reduce the time required to switch tasks by assigning separate tables to each task. Multiple translation tables for multi-task environments can use the Master Enable Flag to enable the appropriate MMUs through software. Multiple translation tables may also be used to extend the physical memory size beyond 16 megabytes by separating system from normal memory and/or program from data memory. The MST and NMS flags in the Mode Register can be used in conjunction with the N/$\overline{S}$ line to select the MMU that contains the appropriate table. Special external circuitry that monitors the CPU Status lines can manipulate the MMU N/$\overline{S}$ line to perform this selection.

DMA Operation

Direct memory access operations may occur between Z8001 instruction cycles and can be handled through the MMU. The MMU permits DMA in either the System or Normal mode of operation. For each memory access, the segment attributes are checked and if a violation is detected, Suppress is activated. Unlike a CPU violation that automatically causes Suppress signals to be generated on subsequent memory accesses until the next instruction, DMA violations generate a Suppress only on a per memory access basis.

The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. No Segment Trap Request is ever generated during DMA, hence warning conditions are not signaled. Trap Requests are not issued because the CPU cannot acknowledge such a request.

At the start of a DMA cycle, DMASYNC must go Low for at least two clock cycles, indicating to the MMU the beginning of a DMA cycle. A Low DMASYNC inhibits the MMU from using an indeterminate segment number on lines SN_0-SN_6. When the DMA logical memory address is valid, the DMASYNC line must be High before a rising edge of Clock and the MMU then performs its address translation and access protection functions. Upon the release of the bus at the termination of the DMA cycle the DMASYNC line must again be High. After two clock cycles of DMASYNC High, the MMU assumes that the CPU has control of the bus and that subsequent memory references are CPU accesses. The first instruction fetch occurs at least two cycles after the CPU regains control of the bus. During CPU cycles, DMASYNC should always be High.

MMU Commands

The various registers in the MMU can be read and written using Z8001 CPU special I/O commands. These commands have machine cycles that cause the Status lines to indicate an SIO operation is in progress. During these machine cycles the MMU enters command mode. In this mode, the rising edge of the Address Strobe indicates a command is present on the AD_8-AD_{15}. If Chip Select is asserted and if this command indicates that data is to be written into one of the MMU registers, the data is read from AD_8-AD_{15} while Data Strobe is Low. If the command indicates that data is to be read from one of the MMU registers, the data is placed on AD_8-AD_{15} while Data Strobe is Low.

There are ten commands that read or write various fields in the Segment Descriptor Register. The status of the Read/Write line indicates whether the command is a read or a write.

The auto-incrementing feature of the Segment Address Register (SAR) can be used to block load segment descriptors using the repeat forms of the Special I/O instructions. The SAR is autoincremented at the end of the field. In accessing the base field, first the high-order byte is selected and then the low-order byte. The command accessing the entire Descriptor Register references the fields in the order of base address, limit and attribute.

Opcode (Hex)	Instruction
08	Read/Write Base Field
09	Read/Write Limit Field
0A	Read/Write Attribute Field
0B	Read/Write Descriptor (all fields)
0C	Read/Write Base Field; Increment SAR
0D	Read/Write Limit Field; Increment SAR
0E	Read/Write Attribute Field; Increment SAR
0F	Read/Write Descriptor; Increment SAR
15	Set All CPU-Inhibit Attribute Flags
16	Set All DMA-Inhibit Attribute Flags

Three commands are used to read and write the control registers.

Opcode (Hex)	Instruction
00	Read/Write Mode Register
01	Read/Write Segment Address Register
20	Read/Write Descriptor Selector Counter Register

The Status Registers are read-only registers, although the Violation Type Register (VTR) can be reset. Nine instructions access these registers.

Opcode (Hex)	Instruction
02	Read Violation Type Register
03	Read Violation Segment Number Register
04	Read Violation Offset (High-byte) Register
05	Read Bus Status Register
06	Read Instruction Segment Number Register
07	Read Instruction Offset (High-byte) Register
11	Reset Violation Type Register
13	Reset SWW Flag in VTR
14	Reset FATL Flag in VTR

MMU Timing

The Z8010 translates addresses and checks for access violations by stepping through sequences of basic clock cycles corresponding to the cycle structure of the Z8001 CPU. The following timing diagrams show the relative timing relationships of MMU signals during the basic operations of memory read/write and MMU control commands. For exact timing information, refer to the composite timing diagram.

Memory Read and Write. Memory read and instruction fetch cycles are identical, except for the status information on the ST_0-ST_3 inputs. During a memory read cycle (Figure 9) the 7-bit segment number is input on SN_0-SN_6 one clock period earlier than the address offset; a High on DMASYNC during T_3 indicates that the segment offset data is valid. The most significant eight bits of the address offset are placed on the AD_0-AD_{15} inputs early in the

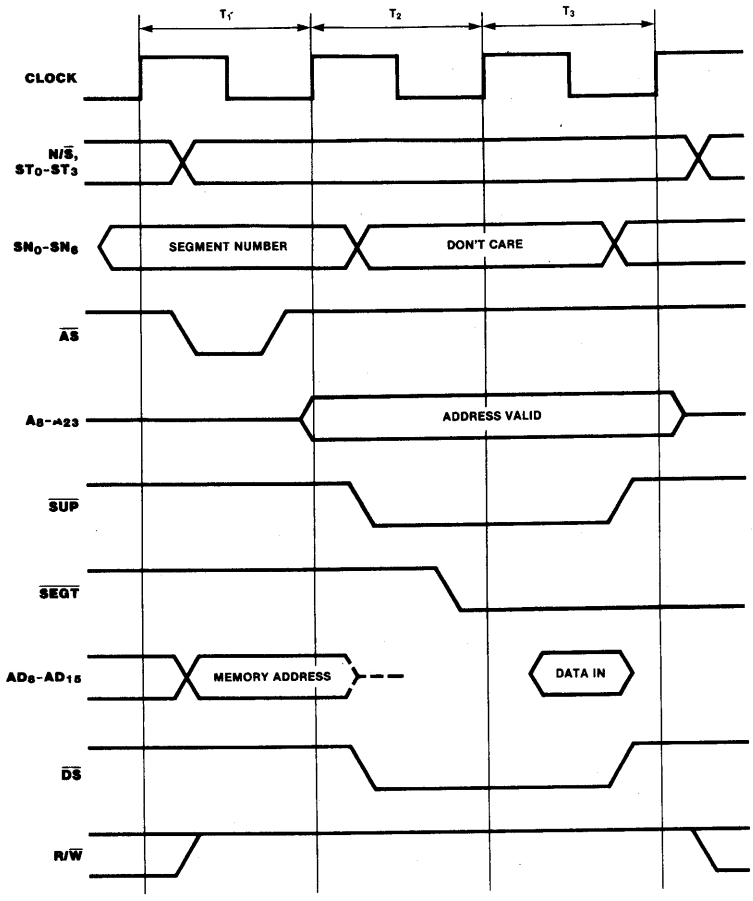


Figure 9. Memory Read Timing

MMU Timing (Continued)

first clock period. Valid address offset data is indicated by the rising edge of Address Strobe. Status and mode information become valid early in the memory access cycle and remain stable throughout. The most significant 16-bits of the address (physical memory location) remain valid until the end of T_3. Segment Trap Request and Suppress are asserted in T_2. Segment Trap Request remains Low until Segment Trap Acknowledge is received. Suppress is asserted during the current machine cycle and terminates during T_3. Suppress is repeatedly asserted during CPU instruction execution cycles until the current instruction has terminated.

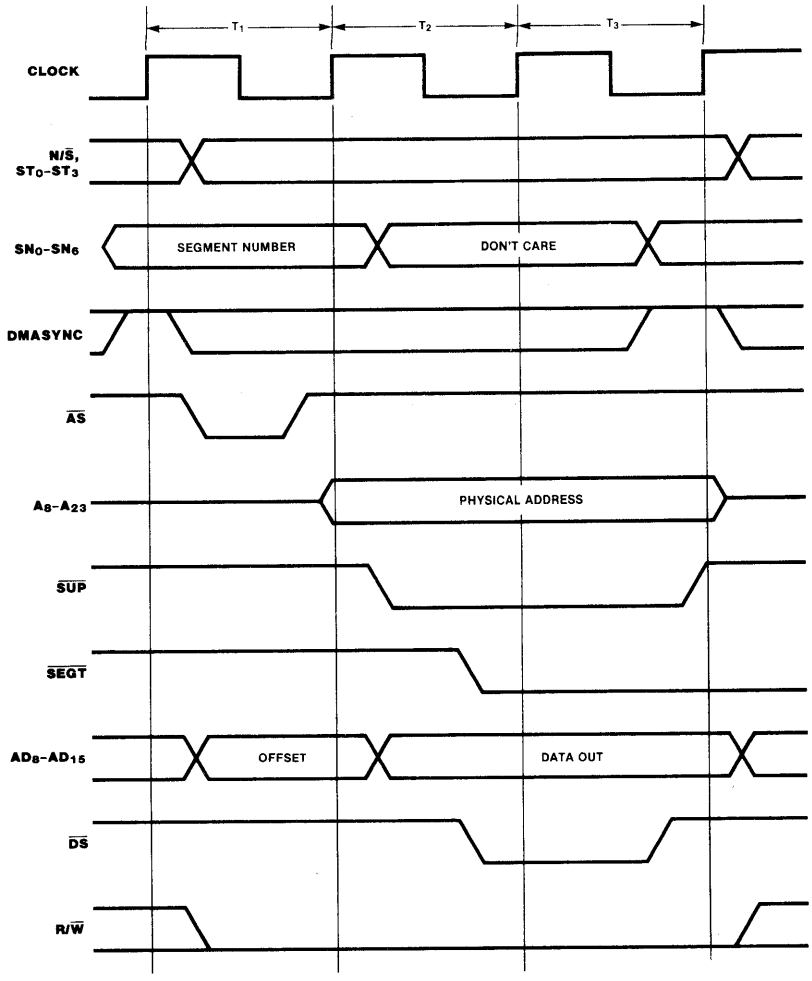


Figure 10. Memory Write Timing

MMU Timing (Continued)

MMU Command Cycle. During the command cycle of the MMU (Figure 11), commands are placed on the Address/Data lines during T_1. The Status lines indicate that a Special I/O instruction is in progress, and the Chip Select line enables the appropriate MMU for that command. Data to be written to a register in the MMU must be valid on the Address/Data lines late in T_2. Data read from the MMU is placed on the Address/Data lines late in the T_{WA} cycle.

Input/Output and Refresh. Input/Output and Refresh operations are indicated by the status lines ST_0-ST_3. During these operations, the MMU refrains from any address translation or protection checking. The address lines A_8-A_{23} remain 3-stated.

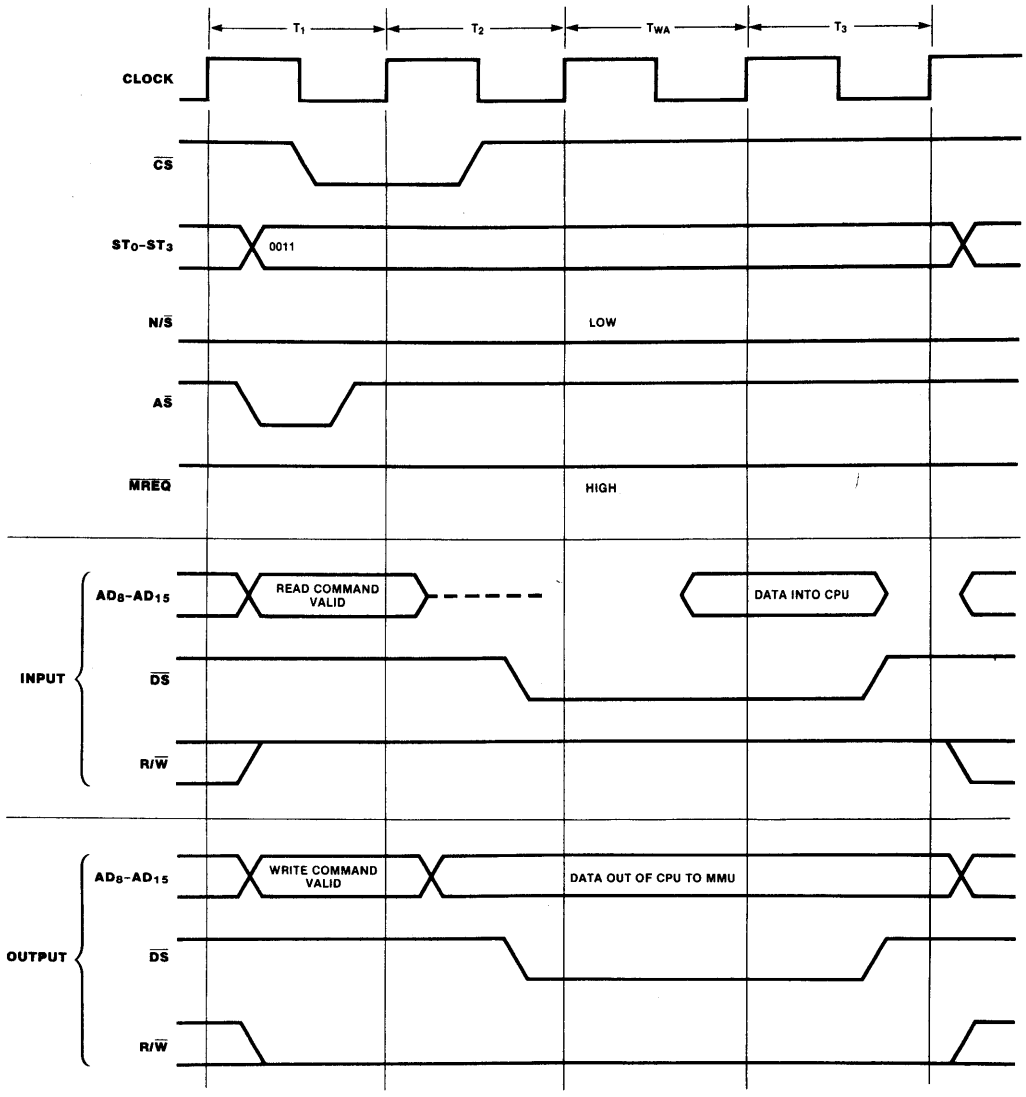


Figure 11. I/O Command Timing

MMU Timing (Continued)

Reset. The MMU can be reset by either hard ware or software mechanisms. A hardware reset occurs on the falling edge of the Reset signal; a software reset is performed by a Z8000 Special I/O command. A hardware reset clears the Mode Register, Violation Type Register and Descriptor Selection Counter. If the Chip Select line is Low, the Master Enable Flag in the Mode Register is set to 1. All other registers are undefined. After reset, the $\overline{AD_8}$-AD_{15} and A_8-A_{23} lines are 3-stated. The $\overline{SUP}$and $\overline{SEGT}$ open-drain outputs are not driven. If the Master Enable flag is not set during reset, the MMU does not respond to subsequent addresses on its A/D lines. To enable an MMU after a hardware reset, an MMU command must be used in conjunction with the Chip Select line.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Violation Type Register and returns the MMU to its initial state (as if no violations or warnings had occurred). Note that the hardware and software resets have different effects.

Segment Trap and Acknowledge. The Z8010 MMU generates a segment trap whenever it detects an access violation or a write into the lowest block of a segment with the DIRW flag set. In the case of an access violation, the MMU also activates Suppress. This Suppress signal can be used to inhibit memory writes. The Segment Trap remains Low until a Trap Acknowledge signal is received. If a violation occurs, Suppress is asserted for that cycle and all subsequent CPU cycles until the end of the instruction; intervening DMA cycles are not suppressed, however, unless they generate a violation. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only, but no Trap Request is generated.

When the MMU issues a Segment Trap Request it awaits a Segment Trap Acknowledge. Subsequent violations occurring before the Trap Acknowledge is received are still detected and handled appropriately. During the Segment Trap Acknowledge cycle, the MMU drives one of its Address/Data lines High; the particular line selected is a function of the identification field of the mode register. After the Segment Trap has been acknowledged by the Z8001/3 CPU, the Violation Status Register should be read via the Special I/O commands in order to determine the cause of the trap. The Trap Type Register should also be reset so that subsequent traps will be recorded correctly.

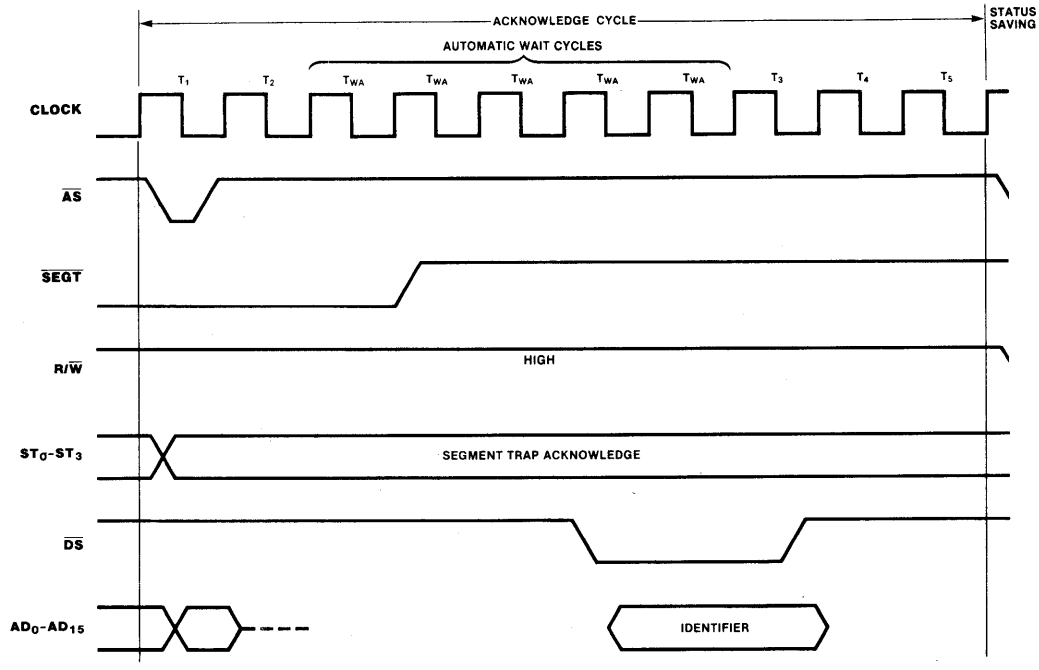


Figure 12. Segment Trap and Acknowledge Timing

Pin Description

A_8-A_{23}. *Address Bus* (outputs, active High, 3-state). These address lines are the 16 most-significant bits of the physical memory location.

AD_8-AD_{15}. *Address/Data Bus* (inputs/outputs, active High, 3-state). These multiplexed address and data lines are used both for commands and for logical addresses intended for translation.

$\overline{AS}$. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ indicates that AD_0-AD_{15}, ST_0-ST_3, R/$\overline{W}$ and N/$\overline{S}$ are valid.

CLK. *System Clock* (input). CLK is the 5 V single-phase time-base input used for both the CPU and MMU.

$\overline{CS}$. *Chip Select* (input, active Low). This line selects an MMU for a control command.

DMASYNC. *DMA/Segment Number Synchronization Strobe* (input, active High). A Low on this line indicates that the segment number lines are 3-state; a High indicates that the segment number is valid. It must always be High during CPU cycles. If a DMA device does not use the MMU for address translation, the $\overline{BUSACK}$ signal from the CPU may be used as an input to DMASYNC.

$\overline{DS}$. *Data Strobe* (input, active Low). This line provides timing for the data transfer between the MMU and the Z8001/3 CPU.

N/$\overline{S}$. *Normal/System Mode* (input, Low = System Mode). N/$\overline{S}$ indicates the Z8001/3 CPU or Z8016 DMA is in the Normal or System Mode. The signal can also be used to switch between MMUs during different phases of an instruction.

Reserved. Do not connect.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the MMU.

R/$\overline{W}$. *Read/Write* (input, Low = write). R/$\overline{W}$ indicates the Z8001/3 CPU or Z8016 DTC is reading from or writing to memory or the MMU.

$\overline{SEGT}$. *Segment Trap Request* (output, active Low, open drain). The MMU interrupts the Z8001/3 CPU with a Low on this line when the MMU detects an access violation or write warning.

SN_0-SN_6. *Segment Number* (inputs, active High). The SN_0-SN_5 lines are used to address one of 64 segments in the MMU; SN_6 is used to selectively enable the MMU.

ST_0-ST_3. *Status* (inputs, active High). These lines specify the Z8001/3 CPU status.

ST_3-ST_0	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference (e.g., to an MMU)
0 1 0 0	Segment trap acknowledge
0 1 0 1	Nonmaskable interrupt acknowledge
0 1 1 0	Nonvectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (EPU)
1 0 1 1	Stack memory request (EPU)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	Extension processor transfer
1 1 1 1	Bus Lock, Data Memory Request (Z8003 only)

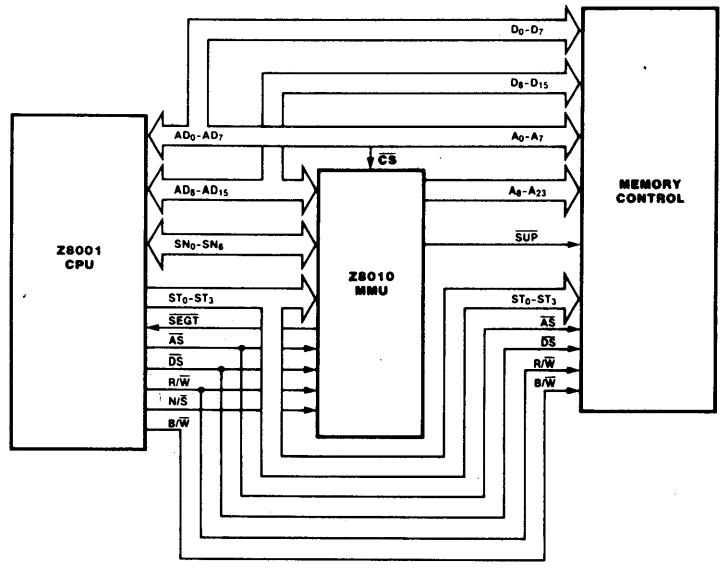


Figure 13. The MMU in a Z8001 System

Pin Description (Continued)

$\overline{SUP}$. *Suppress* (output, active Low, open drain). This signal is asserted during the current bus cycle when any access violation except write warning occurs.

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See ordering information

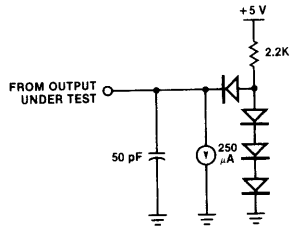
Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- $0°C \leq T_A \leq +70°C$



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		300	mA	

NOTE: The on-chip back-bias voltage generator takes approximately 20 ms to pump the back-bias voltage to -2.5 V after the power has been turned on. The performance of the Z8010 Z-MMU is not guaranteed during this period.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8010	CE	4.0 MHz	Z-MMU (48-pin)	Z8010A	CE	6.0 MHz	Z-MMU (48-pin)
Z8010	CM	4.0 MHz	Same as above	Z8010A	CM	6.0 MHz	Same as above
Z8010	CMB	4.0 MHz	Same as above	Z8010A	CMB	6.0 MHz	Same as above
Z8010	CS	4.0 MHz	Same as above	Z8010A	CS	6.0 MHz	Same as above
Z8010	PE	4.0 MHz	Same as above	Z8010A	PE	6.0 MHz	Same as above
Z8010	PS	4.0 MHz	Same as above	Z8010A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

AC Characteristics

No.	Symbol	Parameter	Z8010 4 MHz Min	Z8010 4 MHz Max	Z8010 6 MHz Min	Z8010 6 MHz Max	Z8010 10 MHz Min	Z8010 10 MHz Max	Notes*†
1	TcC	Clock Cycle Time	250		165		100		
2	TwCh	Clock Width (High)	105		70		40		
3	TwCl	Clock Width (Low)	105		70		40		
4	TfC	Clock Fall Time		20		10		10	
5	TrC	Clock Rise Time		20		15		10	
6	TdDSA(RDv)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		100		80		60	1
7	TdDSA(RDf)	$\overline{DS}$ ↑ (Acknowledge) to Read Data Float Delay		75		60		45	1
8	TdDSR(RDv)	$\overline{DS}$ ↓ (Read) to AD Output Driven Delay		100		80		60	1
9	TdDSR(RDf)	$\overline{DS}$ ↑ (Read) to Read Data Float Delay		75	60	60		45	1
10	TdC(WDv)	CLK ↑ to Write Data Valid Delay		125		80		50	
11	ThC(WDn)	CLK ↓ to Write Data Not Valid Hold Time	30		20		10		
12	TwAS	Address Strobe Width	60		50		30		
13	TsOFF(AS)	Offset Valid to $\overline{AS}$ ↑ Setup Time	45		35		20		
14	ThAS(OFFn)	$\overline{AS}$ ↑ to Offset Not Valid Hold Time	60		40		20		
15	TdAS(C)	AS ↓ to CLK ↑ Delay	110		90		50		
16	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		30		15		
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	50		40		30		
18	TsSN(C)	SN Data Valid to CLK ↑ Setup Time	100		40		20		
19	ThC(SNn)	CLK ↑ to SN Data not Valid Hold Time	0		0		0		
20	TdDMAS(C)	DMASYNC Valid to CLK ↑ Delay	120		80		60		
21	TdSTNR(AS)	Status (ST_0-ST_3, N/$\overline{S}$, R/$\overline{W}$) Valid to $\overline{AS}$ ↑ Delay	50		30		10		
22	TdC(DMA)	CLK ↑ to DMASYNC ↓ Delay	20		15		10		
23	TdST(C)	Status (ST_0-ST_3) Valid to CLK ↑ Delay	100		60		30		
24	TdDS(STn)	$\overline{DS}$ ↑ to Status Not Valid Delay	0		0		0		
25	TdOFF(Av)	Offset Valid to Address Output Valid Delay		175		90		60	1
26	TdST(Ad)	Status Valid to Address Output Driven Delay		155		75		45	1
27	TdDS(Af)	$\overline{DS}$ ↑ to Address Output Float Delay		160		130		100	1
28	TdAS(Ad)	$\overline{AS}$ ↓ to Addres Output Driven Delay		145		70		40	1
29	TdC(Av)	CLK ↑ to Address Output Valid Delay		255		155		100	1
30	TdAS(SEGT)	$\overline{AS}$ ↑ to $\overline{SEGT}$ ↓ Delay		160		100		60	1,2
31	TdC(SEGT)	CLK ↑ to $\overline{SEGT}$ ↑ Delay		300		200		100	1,2
32	TdAS(SUP)	$\overline{AS}$ ↑ to $\overline{SUP}$ ↓ Delay		150		90		55	1,2
33	TdDS(SUP	$\overline{DS}$ ↑ to $\overline{SUP}$ ↑ Delay		155		100		60	1,2
34	TsCS(AS)	Chip Select Input Valid to $\overline{AS}$ ↑ Setup Time	10		10		10		
35	ThAS(CSn)	$\overline{AS}$ ↑ to Chip Select Input Not Valid Hold Time	60		40		20		
36	TdAS(C)	$\overline{AS}$ ↑ to CLK ↑ Delay	0		0		0		
37	TsCS(RST)	Chip Select Input Valid to $\overline{RESET}$ ↑ Setup Time	150		100		60		
38	ThRST(CSn)	$\overline{RESET}$ ↑ to Chip Select Input Not Valid Hold Time	0		0		0		
39	TwRST	$\overline{RESET}$ Width (Low)	2TcC		2TcC		2TcC		
40	TdC(RDv)	CLK ↑ to Read Data Valid Delay		460		300		190	
41	TdDS(C)	$\overline{DS}$ ↑ to CLK ↑ Delay	30		20		10		
42	TdC(DS)	CLK ↓ to $\overline{DS}$ ↑ Delay	0		0		0		

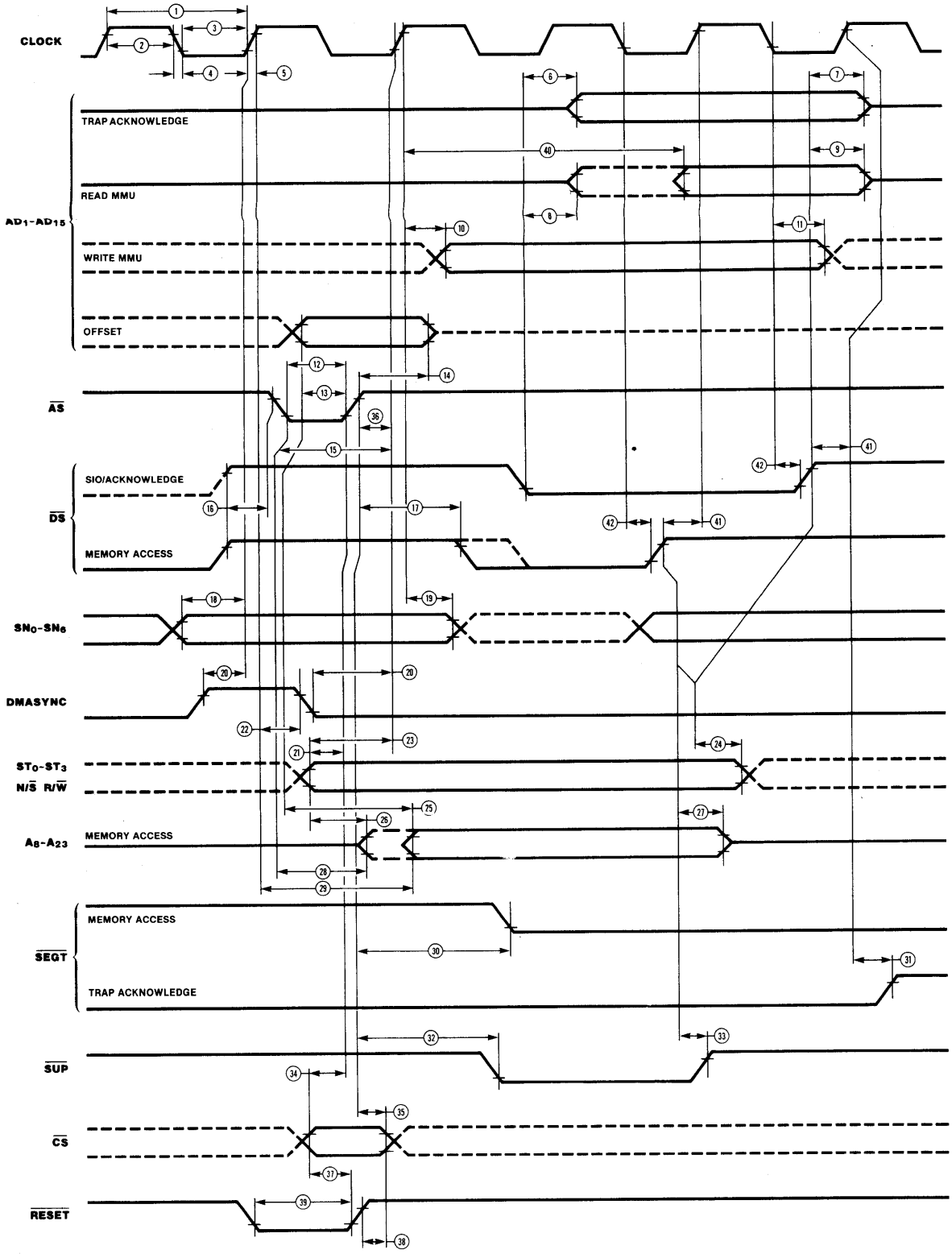
NOTES:
1. 50 pF Load.
2. 2.2K Pull-up.

* All 6 MHz timings are preliminary.

† Units in nanoseconds (ns).

Timing measurements are made at the following voltages:

	High	*Low*
Clock	4.0 V	0.8 V
Output	2.0 V	0.8 V
Input	2.0 V	0.8 V
Float	$\triangle$V	±0.5 V


CLOCK
TRAP ACKNOWLEDGE
READ MMU
AD1-AD15
WRITE MMU
OFFSET
AS
SIO/ACKNOWLEDGE
DS
MEMORY ACCESS
SN0-SN6
DMASYNC
ST0-ST3
N/S R/W
A8-A23
MEMORY ACCESS
SEGT
TRAP ACKNOWLEDGE
SUP
CS
RESET

Z8015 Z8000™ PMMU Paged Memory Management Unit

Zilog

Product Specification

September 1983

FEATURES

- PMMU architecture supports multiprogramming systems and virtual memory implementations.
- Dynamic page relocation makes software addresses independent of physical memory addresses.
- Sophisticated memory management features include access validation that protects memory areas from unauthorized or unintentional access, and a write-warning indicator that predicts stack overflow.
- 64 pages, each 2048 bytes in length, can be mapped into a total physical address space of 16 megabytes; all 64 pages are randomly accessible.
- Pages larger or smaller than 2048 bytes can be easily implemented.
- The number of accessible pages can be increased by using multiple PMMUs.

GENERAL DESCRIPTION

The Z8015 Paged Memory Management Unit (PMMU), a new member of Zilog's Z8000 Family, is designed to support a paged virtual memory system for the Z8003 Virtual Memory Processor Unit (VMPU). Although designed primarily for the Z8003, the PMMU can also be used to support other CPUs in the Z8000 Family. The sophisticated memory management features of the PMMU include access validation for memory protection, a write-warning that gives advance warning of possible stack overflow, and the generation of instruction aborts for accesses to pages not in main memory. Each PMMU can manage a basic memory area of sixty-four 2048-byte, fixed-size pages. The VMPU's 8M byte logical address space is translated by the PMMU into a 16M byte physical address space. Page size can be easily changed and multiple PMMUs can be combined to support more pages.

FUNCTIONAL DESCRIPTION

The Z8015 Paged Memory Management Unit (PMMU) manages the 8M byte addressing spaces of the Z8003 VMPU. The PMMU provides dynamic page relocation as well as numerous memory protection features.

Dynamic page relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is located in the physical memory. It also provides a flexible, efficient method for supporting multiprogramming systems.

The PMMU uses a content-addressable translation table to transform each 23-bit logical address output from the VMPU into a 24-bit address for the physical memory. (Only logical memory addresses go to a PMMU for translation; I/O addresses and data bypass this component.) The translation table consists of 64 page descriptors; each descriptor contains address translation, status, and access information for one memory page. Each PMMU can then manage up to 64 pages of memory.

Multiple PMMUs can be used to support more than 64 pages within a given address space. In addition, PMMUs can be used to accommodate separate translation tables for System and Normal operating modes.

The PMMU is designed to support a memory page 2048 bytes in length. This basic page length can be increased or decreased using a minimal amount of external circuitry.

The PMMU is specially designed to operate with a Z8003 VMPU to implement a paged virtual memory system. If the current PMMU instruction addresses a page not in main memory (a page fault), the PMMU initiates an Instruction Abort operation in the VMPU. During an abort,

the PMMU aborts the execution of the current instruction, then saves the information needed to restart the aborted instruction. On completion of the abort, the PMMU initiates a trap in the VMPU to a routine that brings the addressed page into main memory, updates the descriptor table of the PMMU to allow address translation to the new page, and restarts the execution of the interrupted instruction.

The logical address that caused the page fault is available in three violation address registers of the PMMU. This information can be used to fetch the required instruction or data page into main memory and/or to create a page descriptor entry so that the executing program can access those instructions or data. The logical address of the instruction generating the page fault is available in three instruction address registers of the PMMU. This information can be used to reset the Program Counter to restart the instruction. The instruction to be restarted must also be examined to determine if adjustments must be made to any VMPU registers to ensure correct execution. Finally, the Read/Write Data Count register can be accessed so that certain instructions, such as Load Multiple, can be restarted correctly.

As an aid in implementing efficient paging algorithms, the PMMU provides Changed and Referenced flags for each page descriptor register. The Changed flag indicates that a page has been altered and hence must be copied to secondary storage before that physical memory can be overwritten by another page. The Referenced flag can be used to determine which pages have not been accessed by an executing program—these are the pages that should first be removed from memory when room must be made to bring another page into memory.

PMMU memory protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each page. A page is assigned a number of attributes when its descriptor is initially entered into the PMMU. When a memory reference is made, these attributes are checked against the status information supplied by the VMPU. If a mismatch occurs, the instruction is aborted, a Trap Request signal is generated, and the VMPU is interrupted. The VMPU then checks the status registers of the PMMU to determine the cause of the abort.

Pages are protected by modes of permitted use, such as read only, system only, and execute only. A Valid flag indicates whether or not a descriptor has been initialized. Other page management features include a Write Warning flag useful for stack operations.

The PMMU is controlled by 20 Special I/O instructions, which can be issued from the VMPU in System mode only. With these instructions, system software can assign program pages to arbitrary memory locations, restrict the use of pages, and monitor whether pages have been read or written.

The PMMU has two operating modes: an Address Translation mode in which addresses are translated automatically as they are received, and a Command mode, during which specific registers in the PMMU are accessed using Special I/O commands. Figure 1 shows two simplified block diagrams that illustrate the internal organization and data/signal flow within the PMMU. The resources used in the Translation and the Command modes are shown, separately, in Figures 1a and 1b.

SEGMENTED ADDRESSING AND ADDRESS TRANSLATION

Compared with linear addressing, a segmented addressing space is closer to the way a programmer uses memory because each procedure and data set can reside in its own segment.

The 23-bit addresses output by the VMPU divide an 8M byte addressing space into 128 segments of up to 64K bytes each. A 23-bit segmented address consists of a 7-bit segment number and a 16-bit offset used to address any byte relative to the beginning of the segment. The two parts of the segmented address (segment number and offset) can be manipulated separately.

The PMMU divides physical memory into 2048-byte pages. Pages are assumed to be allocated in memory on 2048-byte boundaries so that the 11 low-order bits of the starting location of each page are always equal to zero.

Segments in a virtual memory system can consist of pages that need not be in physical storage. Those segment pages in main memory need not be contiguous. Segments can have a variable number of pages. Certain pages can be designated so that writes into the last 128 bytes generate a warning trap without causing an instruction abort. If such a page is used as the last page of the system stack, the warning trap can be used to initiate the allocation of another page to the stack segment to prevent a stack overflow error.

The addresses manipulated by the programmer, used by instructions, and output by the VMPU are called logical addresses. The PMMU translates logical addresses into the physical addresses required for accessing memory, this process is called relocation.

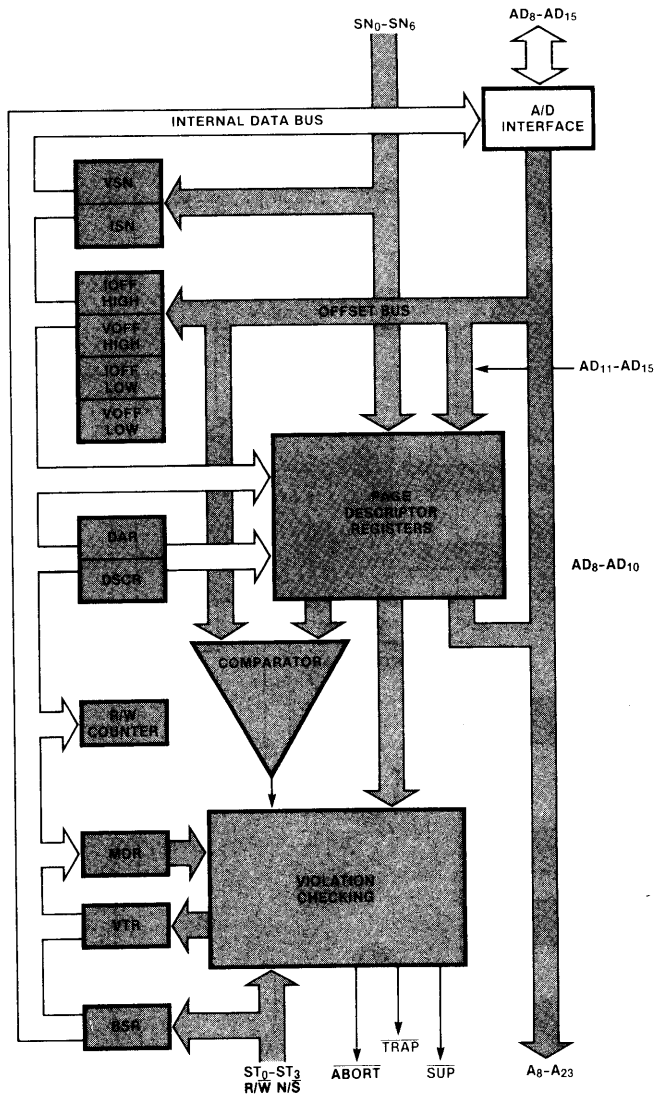


Figure 1a. Address Translation Mode PMMU Operating Modes, Simplified Flow Diagram

The translation activity in the PMMU that provides address relocation is controlled by four internal control flags and six input lines. The four flags are:

Master Enable (MSEN) Flag. This flag controls when the PMMU outputs physical addresses on its Address bus (A) lines. When this bit is clear, the A lines remain 3-stated and no checking is performed.

Translate (TRNS) Flag. This flag determines whether the output on the A lines is the logical address as input (with most significant bit at 0) or a translated address. When translation is not performed, no checking is done.

Multiple Page Table (MPT) Flag. This flag indicates whether or not separate PMMUs are to be used for System and Normal pages.

Normal Mode Select (NMS) Flag. When the Multiple Page Table flag is set, this flag indicates whether the PMMU contains System or Normal page descriptors.

The six input lines used in the control of the PMMU are:

N/$\overline{S}$ Line. This line is used by the PMMU to distinguish System mode accesses from Normal mode accesses. When the Multiple Page Table flag is set, the N/$\overline{S}$ line acts as a chip-select mechanism.

Chip Enable ($\overline{CE}$) Line. This line acts as a master enable control line: it must be asserted for any address translation to occur or for any address to be output by the PMMU.

Status Lines (ST_0-ST_3). These four lines are used by the PMMU to determine the type of transaction in progress.

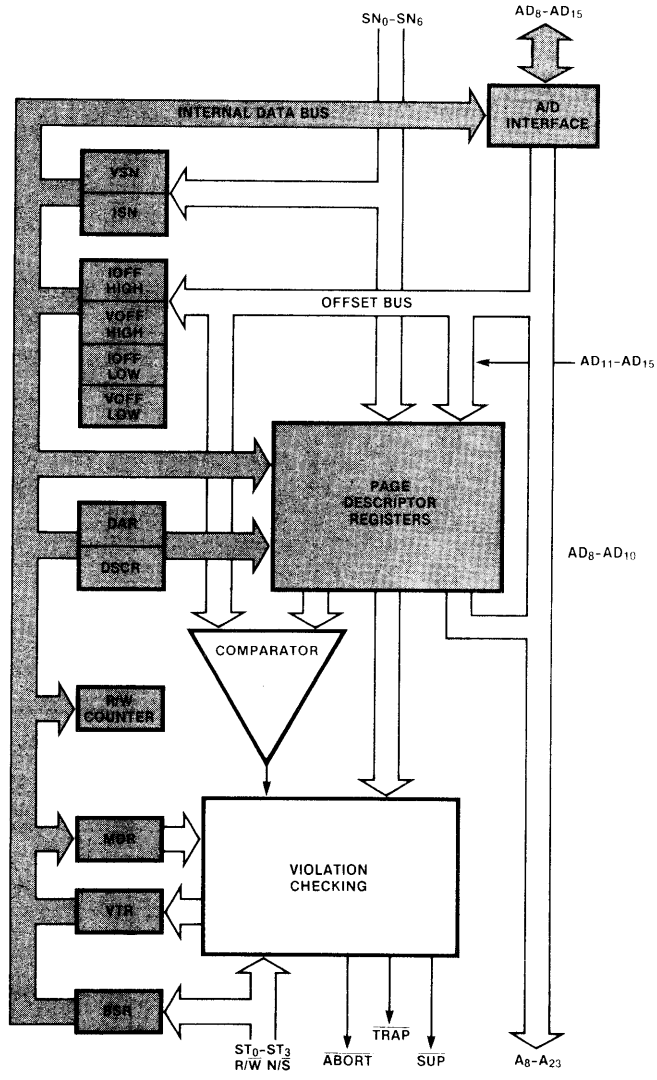


Figure 1b. Command Mode PMMU Operating Modes, Simplified Flow Diagram

Access violation checking, write warning checking, and page fault monitoring functions occur only when the PMMU is enabled for address translation. For example, if Chip Enable is not asserted, the PMMU does not generate an Abort Request even if none of its descriptors match the logical address.

The address translation process is transparent to user software; a simplified flow diagram of this process is shown in Figure 2. A content-addressable translation table in the PMMU compares the 7-bit segment number and five most-significant bits of the offset with the logical address field of each descriptor. If a match occurs and that descriptor's Valid flag is set, the physical address field of that descriptor is accessed. The 11-bit logical address within the page is concatenated to this 13-bit physical base address to obtain the actual physical memory location. Because the base address of a page always has the low-order 11 bits equal to zero, only the high-order 13 bits are stored in the PMMU and used in the translation. The PMMU outputs the 16 most-significant bits of the translated address.

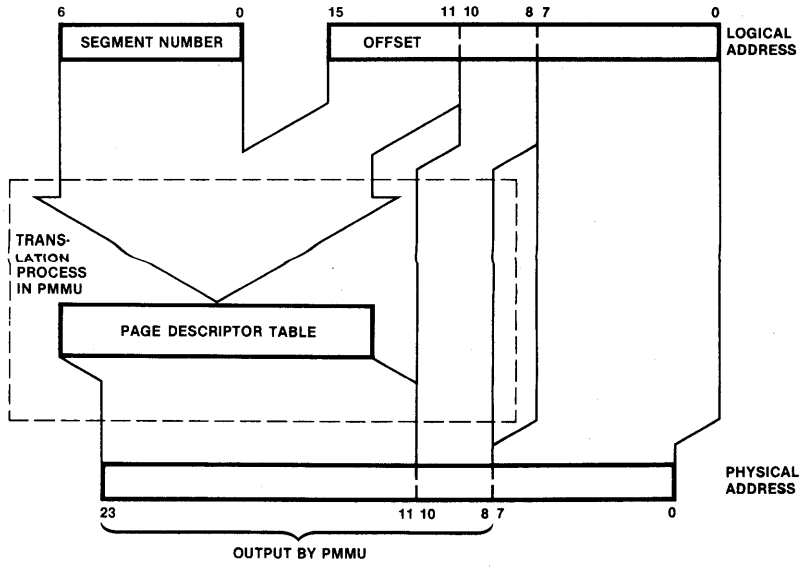


Figure 2. Logical-to-Physical Address Translation

MEMORY PROTECTION

Each memory page is assigned several attributes that are used to provide memory access protection. A memory request from the VMPU is accompanied by status information that indicates the attributes of the memory request. The PMMU compares the memory request attributes with the page attributes and generates Instruction Abort Request, Suppress, and Trap Request signals whenever it detects an attribute violation.

An Abort Request is used to generate the Abort and Wait inputs to the VMPU that cause the current instruction to be aborted. The Suppress input is used by the VMPU to inhibit stores into the memory and thus protect the contents of the memory from erroneous changes. A Trap Request informs the VMPU and the system control program of the violation so that appropriate action can be taken for recovery.

Three attributes: read only, execute only, and system access only, can be associated with each page. When an attempted access violates any one of the page attributes, Abort Request, Trap Request, and Suppress signals are generated by the PMMU.

Each descriptor register has a Valid flag in the attribute field. When set to 1, this flag indicates that the descriptor contains valid translation information and its logical address field is to be used in the associative match process. If Chip Enable is asserted and no match is found, the PMMU, if enabled, generates Abort Request, Trap Request, and Suppress signals. The PMMU is enabled under either of the following conditions: the MSEN and TRNS flags are both 1 and the MPT flag is 0; or the MSEN and TRNS flags are both 1, the MPT flag is 1, and both input N/$\overline{S}$ and the NMS flag have the same value.

Normally, the legal range of offsets within a segment goes from 0 to 65,535 bytes. A stack segment, however, has legal word offsets ranging downward from 65,534 to 0 bytes; the stack manipulation instructions cause stacks to grow toward lower memory locations. When a stack grows to the limit of its allocated segment, additional memory can be added to the segment. As an aid in maintaining stacks, the PMMU detects when a write is performed to the lowest-allocated 128 bytes of a stack page and generates a Trap Request if the DIRW attribute flag is set in the page descriptor. Since neither a Suppress nor Abort Request signal is generated, the write is allowed to proceed. This write warning can then be used to indicate that more memory (that is, another page) needs to be allocated to the segment.

PMMU REGISTER ORGANIZATION

The PMMU contains a set of 64 page descriptor registers that supply the information needed to map logical memory addresses to physical memory locations. The PMMU also contains three 8-bit control registers for programming the PMMU, and nine 8-bit status registers to record information in the event of an access violation.

Page Descriptor Registers

The segment number and five most-significant bits of a logical address determine, by associative lookup, which page descriptor register is used in address translation. Each register also contains the necessary information to enable checking to ensure that the type of reference made is permitted. An indication that the segment has been previously read or written is also contained in the register.

Each of the 64 page descriptor registers contains a 12-bit logical address field, a 13-bit physical address field, and a 7-bit attribute field (Figure 3).

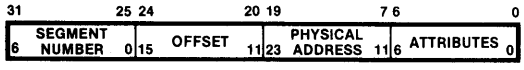


Figure 3. Page Descriptor Register Format

The logical address field is used during the associative search phase of address translation; a match of this field with the most-significant bits of the logical address indicates that the descriptor is to be used during physical address generation. The physical address field supplies the most-significant bits of the generated physical address.

The attribute field contains seven flags (Figure 4). Three flags protect the page against certain types of access, one indicates the special structure of the page, and two encode the types of accesses that have been made to the page. The seventh flag is used to indicate whether or not the information in the descriptor is valid. A flag is set when its value is 1. During a write to only the attribute field, bit 7 of the byte is ignored. When an attribute field is read, bit 7 is undefined. When an entire descriptor is accessed, bit 7 will be part of the physical address field. The following descriptions explain how these flags are used.

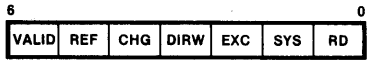


Figure 4. Attribute Field of Page Descriptor Register

Valid (VALID). When this flag is set, the descriptor contains valid page information for the currently executing process. When this bit is clear, the logical address generated by the VMPU is not compared against the contents of the logical address field, so the descriptor is not used for address translation. Only descriptors that have this flag set are used during the associative search.

Read-Only (RD). When this flag is set, the page is read-only and is protected against any write access.

System-Only (SYS). When this flag is set, the page can be accessed only in System mode, and is protected against any access in Normal mode.

Execute-Only (EXC). When this flag is set, the page can be accessed only during an Instruction Fetch cycle and is thus protected against access during other cycles.

Direction and Warning (DIRW). When this flag is set, the page's memory locations are considered to be organized in descending order and each write to the page is checked for access to the lowest 128 bytes. Such an access generates a Trap Request signal to warn of potential stack overflow, but neither an Abort Request nor a Suppress signal is generated.

Changed (CHG). When this flag is set, the page has been changed (written into). This bit is set automatically during any write access to this page if the write access does not cause a violation.

Referenced (REF). When this flag is set, the page has been referenced (either read or written). This bit is set automatically during any access to the page if the access does not cause a violation.

The byte format that is required to write into or read from an attribute field is shown in Figure 5.

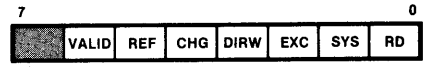


Figure 5. Format of Byte for Reading or Writing a Descriptor's Attribute

Control Registers

The three user-accessible, 8-bit control registers in the PMMU direct the functioning of the PMMU (Figure 6). The Mode register provides a sophisticated method for selectively enabling PMMUs in multiple-PMMU configurations. The Descriptor Address (DAR) register selects the particular page descriptor register to be accessed during a control operation. The Descriptor Selection Count (DSC) register points to the byte in the Page Descriptor register to be accessed during a control operation.

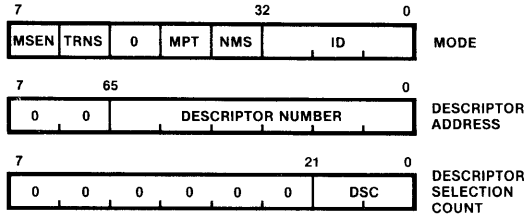


Figure 6. Control Registers

Mode Register

The Mode register contains a 3-bit identification (ID) field that can distinguish up to eight enabled PMMUs in a multiple-PMMU configuration. This field is used during the Trap Request acknowledge sequence. In addition, the Mode register contains the following four flags:

Multiple Page Table (MPT). This flag indicates whether more than one page table is present in the hardware configuration. When this flag is set, more than one table is present and the N/$\overline{S}$ line is used to determine whether the PMMU contains the appropriate table.

Normal Mode Select (NMS). This flag indicates whether the PMMU is to translate addresses when the N/$\overline{S}$ line is High or Low. If the MPT flag is set, the N/$\overline{S}$ line must match the NMS flag for the PMMU to translate addresses; otherwise, the PMMU address lines remain 3-stated.

Translate (TRNS). This flag indicates whether the PMMU is to translate logical program addresses to physical memory locations or is to pass the logical addresses unchanged to the memory without protection checking. In the Non-Translation mode, the most-significant output byte is the 7-bit segment number and the most-significant bit is 0. When TRNS is set, the PMMU performs address translation and attribute checking.

Master Enable (MSEN). This flag enables or disables the PMMU from performing its address translation and memory protection functions. When this flag is set, the PMMU performs these tasks; when the flag is clear the address lines of the PMMU remain 3-stated.

Descriptor Address Register (DAR)

This register points to one of the 64 page descriptor registers. Control commands to the PMMU that access page descriptors implicitly use this pointer to select one of the page descriptor registers. The DAR has auto-incrementing capability so that multiple descriptors can be accessed in a block read/write fashion.

Descriptor Selection Count (DSC) Register

This register holds a 2-bit counter that indicates which byte in the descriptor is being accessed during Read or Write operations. A zero in this counter indicates that the highest-order byte of the descriptor is to be accessed (most-significant byte of the logical address field), a one indicates the next byte of the descriptor, a two indicates the third byte, and a three indicates the least-significant byte (containing the attribute field).

Status Registers

The following nine 8-bit status registers contain information useful in recovering from memory access violations (Figure 7):

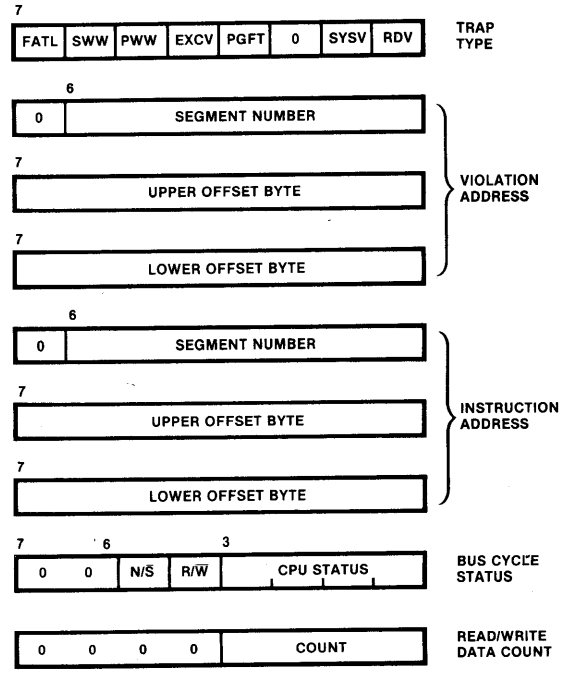


Figure 7. Status Registers

Trap Type Register. This register describes the conditions that generate a Trap Request signal.

Violation Segment Number and Violation Offset Registers. These three registers record the logical address that caused a Trap Request.

Instruction Address Registers. These three registers record the logical address of the last instruction fetched before the first warning, access violation, or page fault.

Bus Cycle Status Register. This register records the bus cycle status (status code, Read/Write operation and Normal/System mode).

Read/Write Data Count Register. This register contains a 4-bit counter that counts the number of read and write data transactions whose addresses have been translated by the PMMU since the last instruction fetch cycle. This count is locked when an Abort Request is generated, and indicates the number of successful data transactions performed by the aborted instruction.

Violation Type Register (VTR) Flags

The VTR is used by the PMMU to determine the cause of a Trap Request. The PMMU generates a Trap Request when: it detects an access violation, such as an attempt to write into a read-only page; it detects a warning condition, which is a write into the lowest 128 bytes of a page with the DIRW flag set; or no entry matches the logical address (a page fault). The following seven flags are contained by the Violation Type register (VTR):

Read-Only Violation (RDV). This flag is set when the VMPU attempts to access a read-only page and the R/$\overline{W}$ line is Low.

System Violation (SYSV). This flag is set when the VMPU accesses a system-only page and the N/$\overline{S}$ line is High.

Execute-Only Violation (EXCV). This flag is set when the VMPU attempts to access an execute-only page other than during an instruction fetch cycle.

Page Fault (PGFT). This flag is set when no logical address field of the valid descriptors in the PMMU matches the upper 12 bits of the logical address.

Primary Write Warning (PWW). This flag is set when an access is made to the lowest 128 bytes of a page with the DIRW flag set.

Secondary Write Warning (SWW). This flag is set when the VMPU writes data into the last 128 bytes of the system stack and EXCV, SYSV, PGFT, RDV, or PWW is set. With SWW set, subsequent write warnings for accessing the system stack do not generate a Trap Request.

Fatal Condition (FATL). This flag is set when any other flag in the Trap Type register is set and either a violation is detected or a write warning condition occurs in Normal mode. FATL is not set during a stack push in System mode that results in a warning condition. This flag indicates that a memory access error has occurred in the trap processing routine. Once set, no Trap Request or Abort Request signals are generated on subsequent violations. However, as long as the PMMU is enabled, Suppress signals are generated on this and subsequent VMPU violations until the FATL flag is reset.

ABORT, TRAP REQUEST, AND ACKNOWLEDGE

The PMMU generates a four-clock-cycle Abort Request ($\overline{ABORT}$) when it detects an access violation or a page fault. This signal on the $\overline{ABORT}$ (pin 33) and $\overline{WAIT}$ (pin 20) inputs of the VMPU inserts a five-cycle abort sequence that causes the VMPU to terminate instruction execution. A Trap Request is generated when the PMMU detects an access violation, page fault, or write warning. This signal on the translation trap line of the VMPU (line 14) causes the VMPU to generate a trap acknowledge after the instruction abortion (for an access violation or page fault) or after the execution of the instruction (for a write warning). In the case of an access violation or page fault, the PMMU also activates Suppress ($\overline{SUP}$), which can be used by the memory to inhibit memory writes.

Trap Request remains Low until a trap acknowledge is received (status = 0100). If a VMPU-generated violation occurs, Suppress is asserted for that memory reference. (If a Z8001 or Z8002 CPU generates the violation, any subsequent CPU memory reference also causes Suppress to be asserted until the end of the instruction.) Intervening DMA accesses are not suppressed, however, unless they generate a violation. Violations detected during DMA accesses cause Suppress to be asserted for that access only; no Trap or Abort Requests are generated during DMA accesses.

Trap Requests to the VMPU are handled similarly to interrupts. To service a PMMU trap, the VMPU issues a trap acknowledge. The acknowledge is usually preceded by a dummy instruction fetch that is not used by the VMPU (the PMMU has been designed to ignore this dummy fetch). During the identifier fetch of the acknowledge cycle, all enabled PMMUs use the Address/Data (AD) lines to indicate their status. A PMMU that has generated a Trap Request outputs a 1 on the AD line associated with the number in its ID field; a PMMU that has not generated a trap request outputs a 0 on its associated AD line. AD lines with no associated PMMU remain 3-stated. During a trap acknowledge, a PMMU uses AD line 8 + i if its ID field is i.

Following the Acknowledge cycle, the VMPU automatically pushes the program status onto the system stack and loads another program status from the trap vector at location 20_H in the program status area. The PMMU's trap line is reset during the trap acknowledge. Suppress is not generated during the stack push. If the push operation creates a write warning condition, a Trap Request is generated and serviced at the end of the context swap. The SWW flag is also set. Servicing this second Trap Request also creates a write warning condition, but because the SWW flag is set, no Trap Request is generated. If a violation or page fault rather than a write warning occurs during the context swap, the FATL flag is set rather than the SWW flag. Subsequent violations or faults cause Suppress but not Trap Request to be asserted. Without the SWW and FATL flags, trap processing routines that generate memory violations or faults would repeatedly be interrupted and called to process the trap they created.

The VMPU routine to process a Trap Request should first check the FATL flag to determine if a fatal system error has occurred. If not, the SWW flag should be checked to determine if more memory is required for the system stack. Finally, the trap itself should be processed and the Trap Type register reset.

MULTIPLE PMMUs

Although only one PMMU should be actively translating addresses at a given time, PMMU architecture directly supports various methods for multiple PMMU configurations. The following four examples illustrate different ways that PMMUs can be used to implement memory management systems capable of handling more than 64 pages.

Example 1. The first approach extends the capability of one PMMU for handling 64 pages to a multiple-PMMU configuration that manages more than 64 pages. The Chip Enable line is used to select a particular PMMU to translate a page address. For example, if one PMMU is assigned only pages for logical addresses whose bit number 11 is a 0 and another PMMU is assigned those whose bit number 11 is a 1, then the state of output line AD_{11} can be used to select the appropriate PMMU to translate a logical address.

Example 2. Another way of using Chip Enable separates program pages from data pages. One PMMU is associated with translating addresses generated during instruction fetches (status codes 1100 and 1101) and the other with addresses generated during data fetches (status codes 1000, 1001, 1010, 1011, and 1111). Chip Enable for each PMMU is obtained from the status code (i.e., status lines ST_0–ST_3).

Example 3. Several PMMUs can be used to implement multiple translation tables. Multiple tables reduce the time required to switch tasks by assigning separate tables to each task. Multiple translation tables for multitask environments can use the Master Enable flag to enable the appropriate PMMUs through software.

Example 4. A final method uses two translation tables to separate system from normal memory. The MPT and NMS flags in the Mode register can be used in conjunction with the N/$\overline{S}$ line to select the PMMU that contains the appropriate table.

CHANGING PAGE SIZE

The PMMU directly supports pages of 2048 bytes in length. However, the addition of external circuitry enables the PMMU to support systems with larger or smaller pages. The following examples illustrate the technique for changing the supported page size.

Example 1. To implement 4096-byte pages, address bit AD_{11} is not used in the translation process but is used directly as the most significant bit of the address location within a selected 4096-byte page. This can be achieved by doing the following: (1) set the AD_{11} input to be equal to the logical OR of the ST_3 and AD_{11} output lines of the VMPU, (2) require that the least significant bit in the logical address field of each descriptor register be set to 1, and (3) use the logical address bit AD_{11} output by the VMPU instead of the physical address A_{11} output by the PMMU. The AD_{11} input must be 1 during address translation but must equal the AD_{11} output by the CPU during command cycles to the PMMU; ST_3 is used to distinguish the two types of transactions.

Example 2. To implement 1024-byte pages an additional address bit must be translated. Two PMMUs are used. The $\overline{CE}$ input is driven by AD_{10} for one PMMU and its complement, $\overline{AD}_{10}$, for the other.

DMA OPERATION

At the start of a DMA cycle, DMASYNC must go Low whether or not the PMMU is used to translate DMA addresses, to indicate the beginning of a DMA cycle. When DMASYNC has been Low for two cycles, the PMMU assumes that a DMA device has control of the bus. A Low on DMASYNC inhibits the PMMU from using an indeterminate segment number on lines SN_0–SN_6. When the DMA logical memory address is valid, the DMASYNC line must be High on a rising edge of the clock and then be Low by the next falling clock edge. The PMMU then performs its address translation and access protection functions during the clock period begun in this DMASYNC pulse. Upon the release of the bus at the termination of the DMA cycle, the DMASYNC line must go High. After two clock cycles of DMASYNC High, the PMMU assumes that the VMPU has control of the bus and that subsequent memory references are VMPU accesses. The first memory reference occurs at least two cycles after the VMPU regains control of the bus. During VMPU cycles, DMASYNC should remain High. Refer to the paragraph "Memory Read and Write" and Figure 8 for further information.

Direct memory access (DMA) operations can occur between instruction cycles and can be handled through the PMMU. The PMMU permits DMA in either the System or Normal mode of operation. For each memory access, the page attributes are checked, and if a violation is detected, Suppress is activated. DMA violations generate a Suppress only on a per-memory-access basis.

The DMA device should note the Suppress signal and record sufficient information to enable the system to recover from the access violation. Neither a Trap Request nor an Abort Request is ever generated during a DMA operation, therefore warning conditions are not signaled.

PMMU COMMANDS

The various registers in the PMMU can be read and written using VMPU Special I/O commands. The machine cycles of these commands cause the status lines to indicate that a special input/output operation is in progress. During these machine cycles, the PMMU enters the command mode. In this mode, the rising edge of $\overline{AS}$ indicates that a command is present on lines AD_8–AD_{15}. If this command indicates that data is to be written into one of the PMMU registers, the data is read from lines AD_8–AD_{15} while $\overline{DS}$ is Low. If the command indicates that data is to be read from one of the PMMU registers, the data is placed on lines AD_8–AD_{15} while $\overline{DS}$ is Low.

There are five commands that read or write various fields in the page descriptor register. The status of the read/write line indicates whether the command is a read or a write.

The autoincrementing feature of the Descriptor Address Register (DAR) can be used to block load page descriptors using the repeat forms of the Special I/O instructions. The DAR is autoincremented at the end of the field. The command accessing the entire page descriptor register references the fields in the order of logical address, physical address, and attribute; four bytes are written in succession.

Table 1 gives the five commands that are used to write data into descriptor fields.

Table 1. Descriptor Field Write Commands

Opcode (Hex)	Instruction
OA	Read/Write Attribute field
OB	Read/Write Descriptor (all fields)
OE	Read/Write Attribute field; increment DAR
OF	Read/Write Descriptor (all fields); increment DAR
15	Reset all Valid Attribute flags

Table 2 gives the three commands that are used to read and write the control registers.

Table 2. Control Registers' Read/Write Commands

Opcode (Hex)	Instruction
00	Read/Write Mode register
01	Read/Write Descriptor Address register
20	Read/Write Descriptor Selector Count register

The status registers are read-only registers, although the Trap Type Register (TTR) can be reset. Twelve instructions, shown in Table 3, access these registers.

Table 3. Status Registers' Access Instructions

Opcode (Hex)	Instruction
02	Read Trap Type register
03	Read Violation Segment Number register
04	Read Violation Offset (high-byte) register
05	Read Bus Status register
06	Read Instruction Segment Number register
07	Read Instruction Offset (high-byte) register
11	Reset Trap Type register
13	Reset SWW flag in VTR
14	Reset FATL flag in VTR
21	Read Violation Offset (low-byte) register
22	Read Read/Write Data Counter register
23	Read Instruction Offset (low-byte) register

USE OF THE PMMU WITH OTHER Z8000 CPUs

The PMMU is designed to operate in conjunction with the Z8003 VMPU; however, it can also be used with other CPUs in the Z8000 Family. The following examples suggest simple system configurations; more sophisticated arrangements are possible.

The Z8004 VMPU generates nonsegmented 16-bit addresses only. The PMMU can be used to implement a paged virtual memory by tying the segment number inputs of the PMMU to 0 and requiring the most significant seven bits of the logical address field to be 0. Since the Z8004 VMPU lacks a translation trap request input pin, the nonmaskable (or other interrupt request) pin should be used instead.

The PMMU extends the physical addressing capability of the Z8004 without using the segmentation mechanism of the Z8003. This use is similar to the way in which 16-bit minicomputers extend their addressing capability.

The Z8001 and Z8002 CPUs do not support the instruction abort mechanism. A page fault for one of these CPUs is, in general, non-recoverable, since during an interrupt, the current instruction runs to completion, possibly overwriting CPU registers. For the nonsegmented Z8002, this means that all pages that the CPU can access must be in physical memory and appropriate information must be in the PMMU page descriptor registers. For the segmented Z8001, this means that programs must explicitly request segments before accessing them and must free segments after use. It also means that segments are allocated in units of the page size and that limit protection is performed with this granularity. Use of the PMMU with the Z8001 and Z8002 CPUs permits a paged allocation of main memory and extends the physical address capability of the Z8002.

PMMU TIMING

The PMMU translates addresses and checks for access violations by stepping through sequences of basic clock cycles corresponding to the cycle structure of the VMPU. Timing diagrams that show the relative timing relationships of PMMU signals during the basic operations of memory read/write and PMMU control commands are given in this section.

Memory Read and Write

Memory read and instruction fetch cycles are identical, except for the status information on the ST_0–ST_3 inputs. During a memory read cycle (Figure 8), the 7-bit segment number is input on SN_0–SN_6 one clock period before the address offset; a High on DMASYNC during T3 indicates that the segment offset data is valid. The address offsets are placed on the AD_0–AD_{15} inputs early in the first clock period. Valid address offset data is indicated by the rising edge of $\overline{AS}$. Status, mode, and chip enable information becomes valid early in the memory access cycle and must remain stable throughout. The most significant 16 bits of the address (physical memory location) remain valid until the end of T3. Abort Request, Trap Request, and Suppress are asserted in T2 (Figure 9). Abort Request is asserted for four clock cycles. Trap Request remains Low until trap acknowledge (status = 0100) is received. Suppress is asserted during the current machine cycle and terminates during T3.

PMMU Command Cycle

During the command cycle of the PMMU (Figure 11), commands are placed on lines AD_8–AD_{15} during T1. The status lines indicate that a Special I/O instruction is in progress, and the $\overline{CS}$ line enables the appropriate PMMU for that command. Data to be written to a register in the PMMU must be valid on lines AD_8–AD_{15} late in T2. Data read from the PMMU is placed on lines AD_8–AD_{15} late in the T_{WA} cycle.

Input/Output and Refresh

Input/output and refresh operations are indicated by codes on status lines ST_0–ST_3. During these operations, the PMMU refrains from any address translation or protection checking. Lines A_8–A_{23} remain 3-stated during these operations.

Reset

The PMMU can be reset by either hardware or software mechanisms. A hardware reset occurs on the falling edge of the Reset signal; a software reset is performed by a VMPU special I/O command. A hardware reset clears the Mode register, Trap Type Register, and Descriptor Selection Count register. If the $\overline{CS}$ line is Low, the Master Enable flag in the Mode register is set to 1. All other registers are undefined. After reset, lines AD_0–AD_{15} and A_8–A_{23} are 3-stated. The $\overline{SUP}$ and $\overline{ABORT}$ open-drain outputs are not driven. If the Master Enable flag is not set during reset, the PMMU does not respond to subsequent addresses on its AD lines. To enable a PMMU after a hardware reset, a PMMU command must be used in conjunction with $\overline{CS}$.

A software reset occurs when the Reset Violation Type Register command is issued. This command clears the Trap Type Register and returns the PMMU to its initial state (as if no violations or warnings had occurred). Note that the hardware and software resets have different effects.

Abort, Trap Request, and Acknowledge

The PMMU generates a Trap Request whenever it fails to find a page entry corresponding to the logical address (that is, a page fault), detects an access violation, or detects a write into the lowest 128-byte block of a page with the DIRW flag set (Figure 12). In the case of an access violation or page fault, the PMMU also activates Suppress and Abort Request. The Suppress signal is used by memory to inhibit memory writes. The Trap Request remains Low until a trap acknowledge signal (status = 0100) is received. Violations detected during DMA cycles cause Suppress to be asserted during that cycle only, but no Trap Request is generated.

When the PMMU issues a Trap Request, it awaits the indication of a trap acknowledge. Subsequent violations occurring before the trap acknowledge indication is received are detected and appropriately processed. During a Trap Acknowledge cycle, the PMMU drives one of its Address/Data lines; the selection of the line is a function of the identification field of the Mode register. After the Trap Request has been acknowledged by the VMPU, the Violation Status register should be read by a special I/O command in order to determine the cause of the trap. The Trap Type register should be reset so that subsequent traps are recorded correctly.

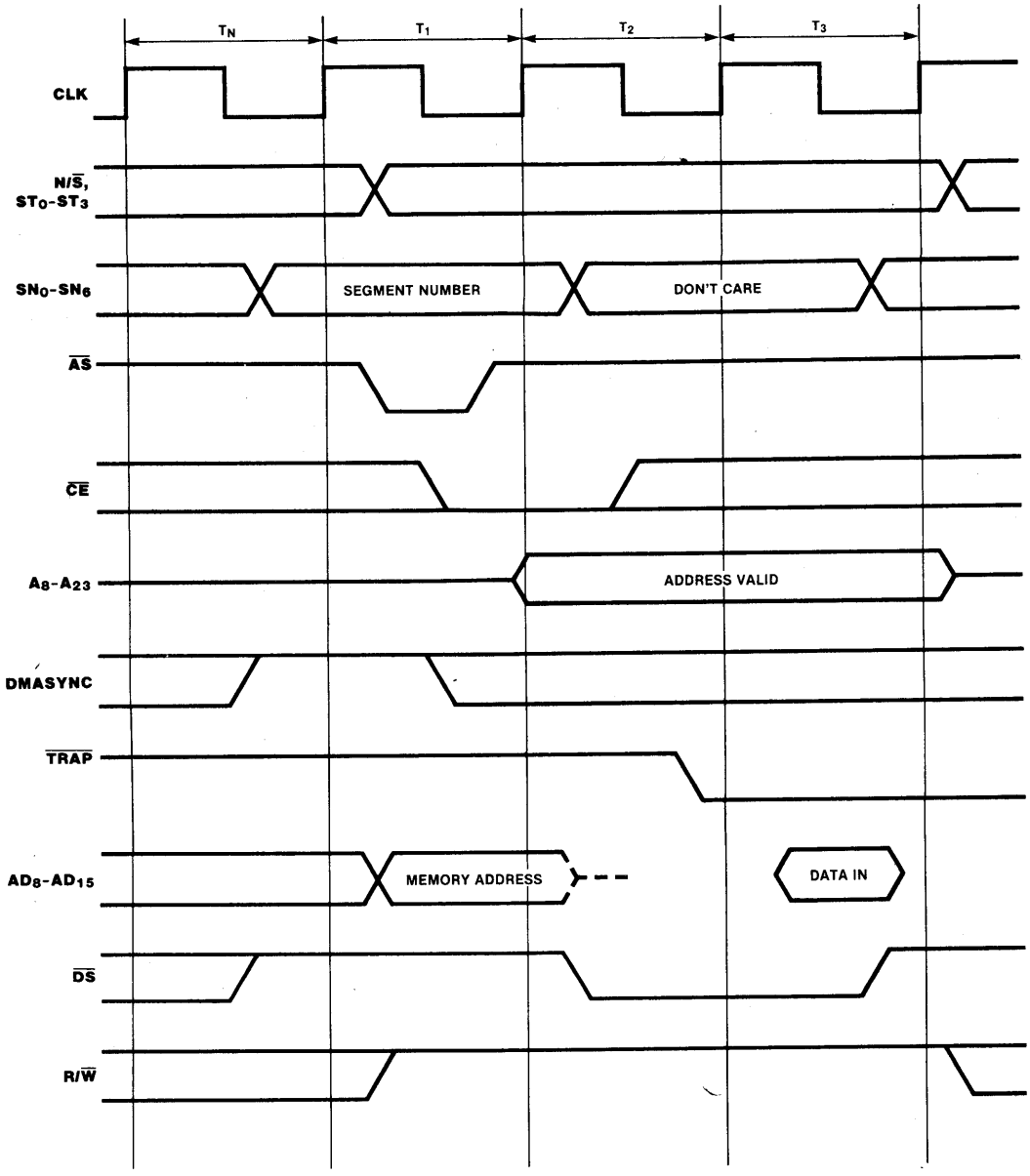


Figure 8. Memory Read Timing

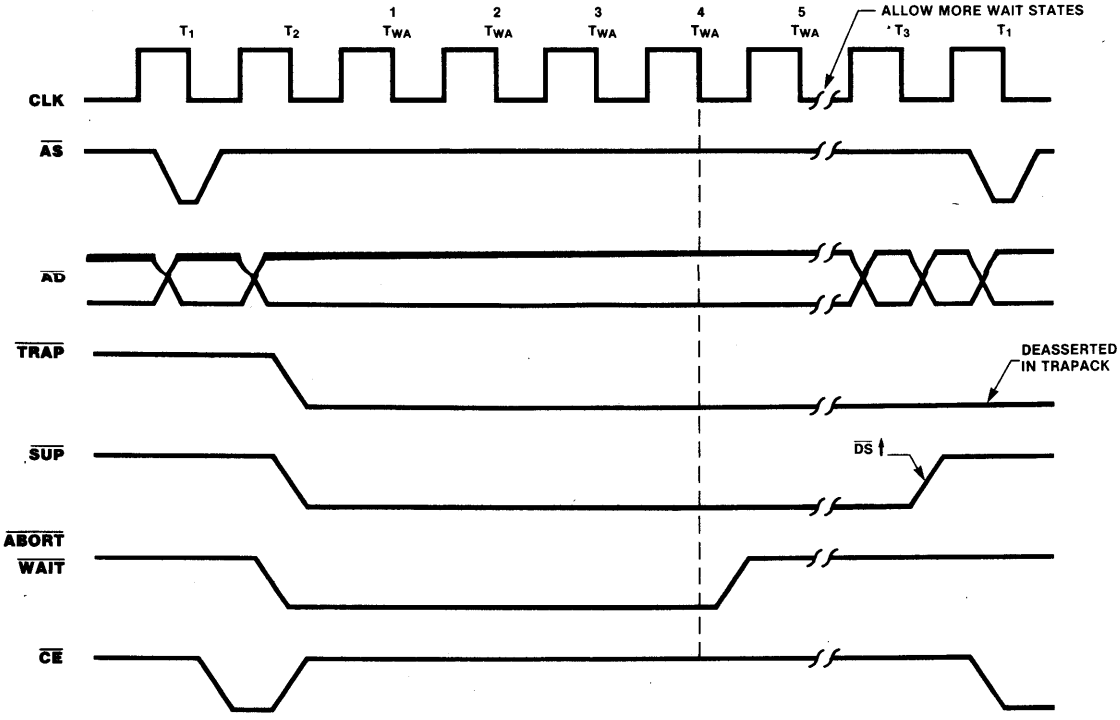


Figure 9. Abort and Trap Request Timing

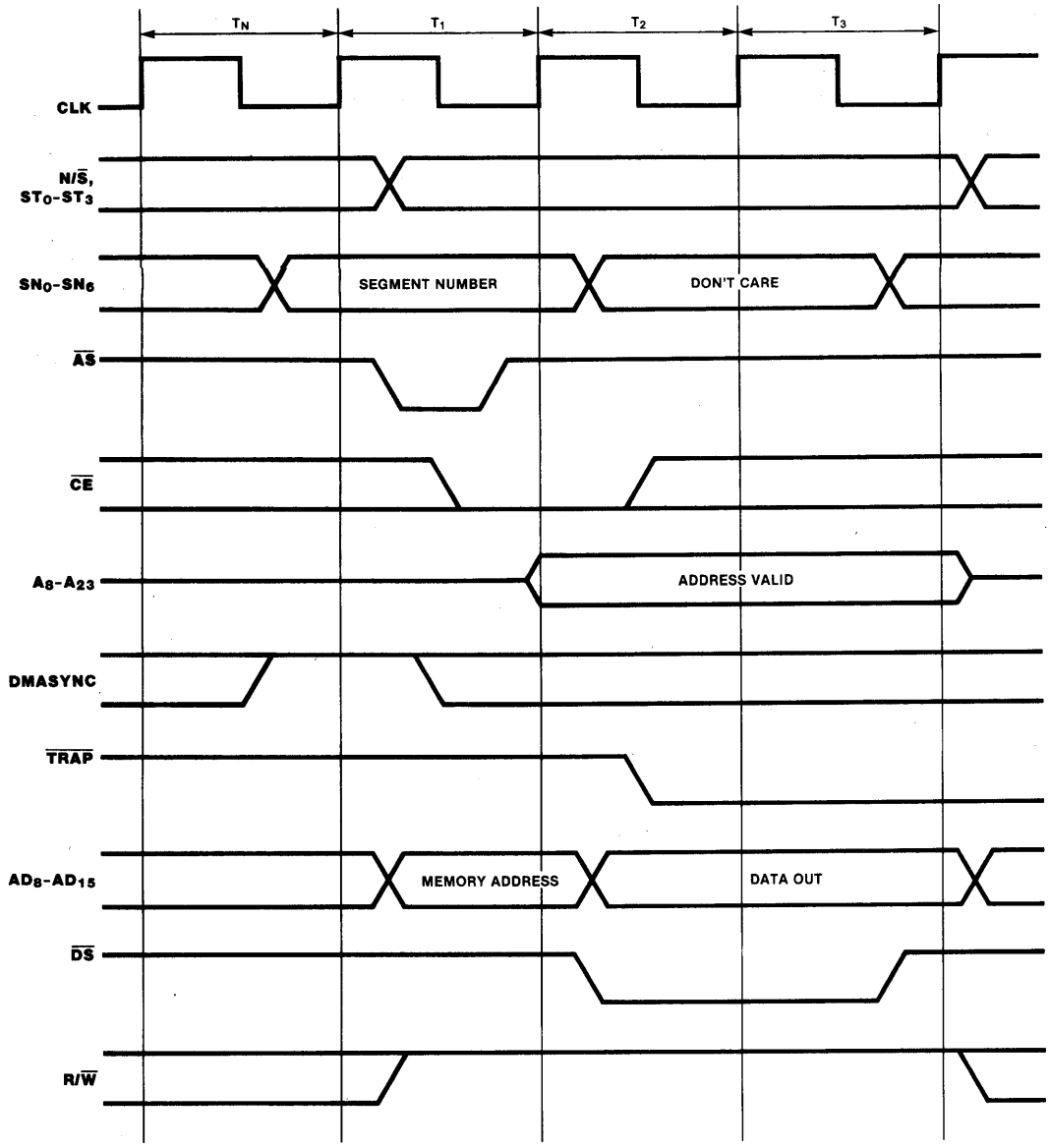


Figure 10. Memory Write Timing

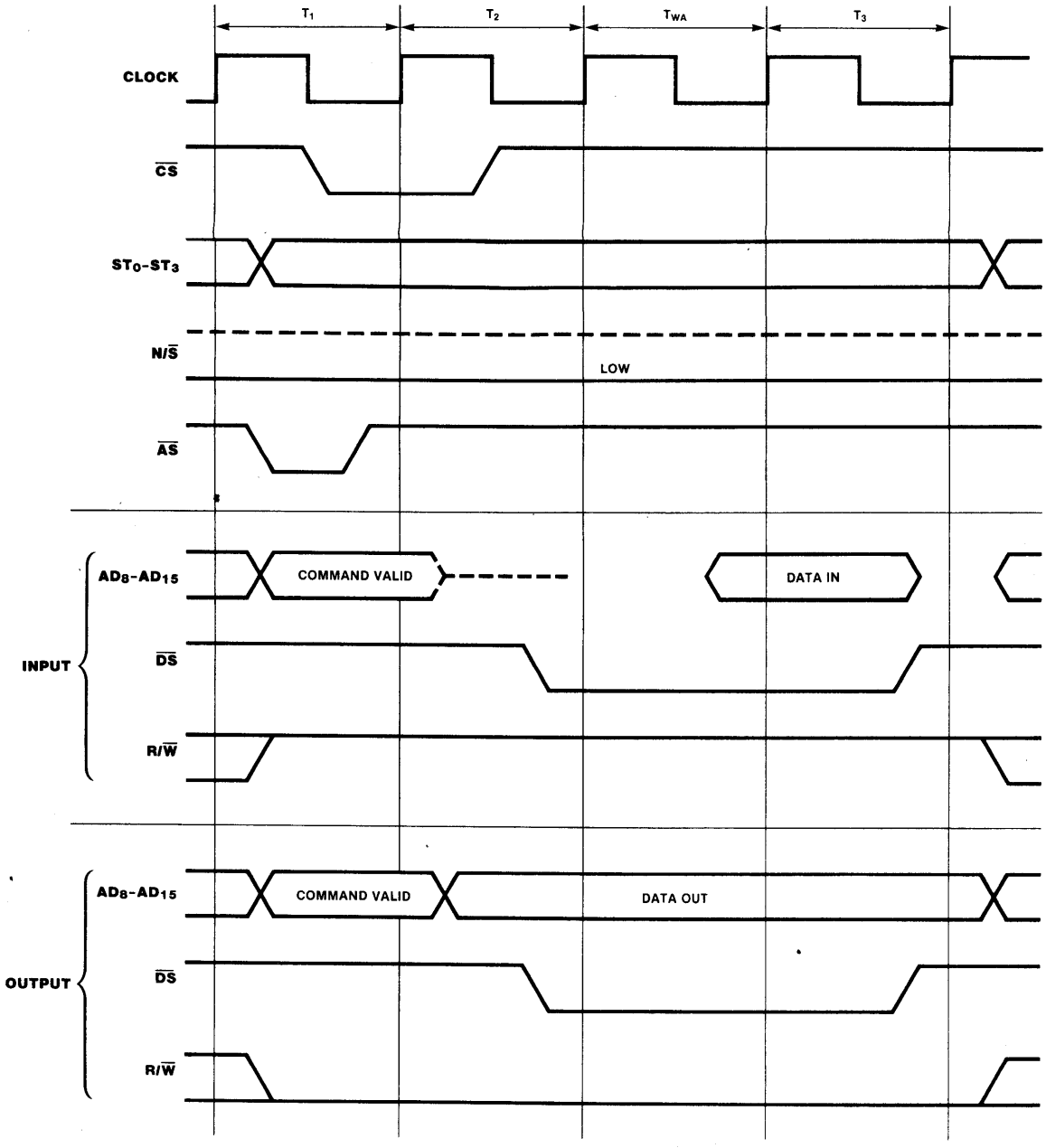


Figure 11. I/O Command Timing

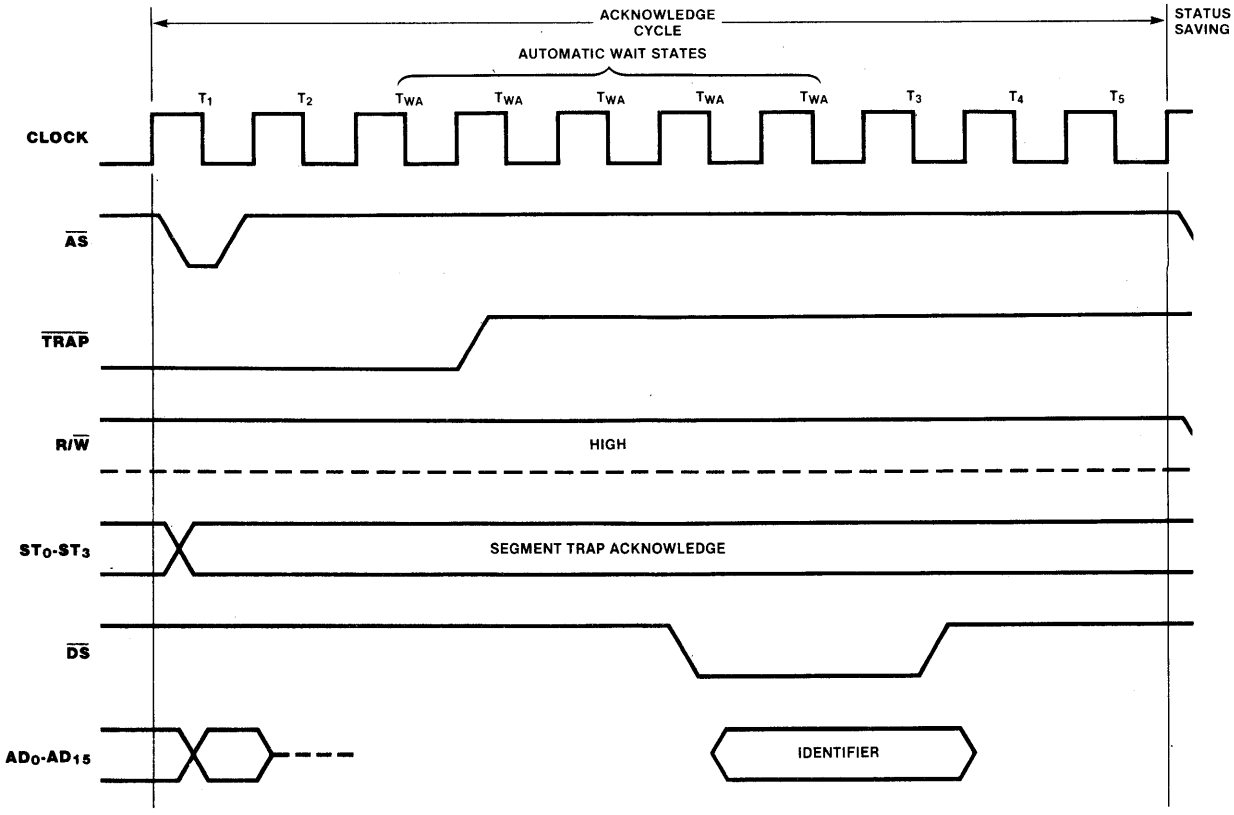


Figure 12. Trap Request and Acknowledge Timing

SIGNAL DESCRIPTIONS

The Z8015 is produced in a 64-pin package; the functions performed by the device's input and/or output pins are shown in Figure 13. Pin/signal name assignments are shown in Figure 14. The device signal names together with a brief description of the function(s) performed by each are given in the following paragraphs.

A_8-A_{23}. *Address Bus* (outputs, active High, 3-state). These address lines are the 16 most significant bits of the physical memory location.

$\overline{ABORT}$. *Abort Request* (output, active Low, open drain). A Low on this line indicates MMU requests for an instruction abort. This line is enabled when a page fault or access violation is detected.

AD_0-AD_{15}. *Address/Data Bus* (inputs/outputs, active High, 3-state). AD_0–AD_7 are used for addresses and inputs only. They carry the low-order byte in the offset of logical addresses intended for translation. AD_8–AD_{15} are multiplexed address and data lines that are used both for the eight most significant bits of the logical address and for commands.

$\overline{AS}$. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ indicates that lines AD_0–AD_{15}, ST_0–ST_3, R/$\overline{W}$, $\overline{CE}$, and N/$\overline{S}$ are valid.

$\overline{CE}$. *Chip Enable* (input, active Low). This line selects a PMMU to translate a logical address.

CLK. *System Clock* (input). CLK is a +5 V single-phase, time-base input used for both the VMPU and PMMU.

$\overline{CS}$. *Chip Select* (input, active Low). This line selects a PMMU for a control command.

DMASYNC. *DMA/Segment Number Synchronization Strobe* (input, active High). A Low on this line indicates a DMA access is occurring; a High indicates the segment number is valid. It must be High during VMPU cycles and Low when SN lines are 3-stated.

$\overline{DS}$. *Data Strobe* (input, active Low). This line provides timing for the data transfer between the PMMU and the Z8003 VMPU.

N/$\overline{S}$. *Normal/System Mode* (input, Low = System mode). N/$\overline{S}$ indicates to the PMMU that the VMPU or DMA is in the Normal or System mode.

$\overline{RESET}$. *Reset* (input, active Low). A Low on this line resets the PMMU.

R/$\overline{W}$. *Read/Write* (input, Low = write). R/$\overline{W}$ indicates whether the VMPU is reading from or writing to either memory or the PMMU.

SN_0-SN_6. *Segment Number* (inputs, active High). These lines provide the 7-bit segment number of a logical address.

ST_0-ST_3. *Status* (inputs, active High). These lines (Table 4) specify the status of the associated VMPU.

$\overline{SUP}$. *Suppress* (output, active Low, open-drain). This signal is asserted during the current bus cycle when a page fault or any access violation, except write warning, occurs.

$\overline{TRAP}$. *Trap Request* (output, active Low, open-drain). The PMMU interrupts the VMPU with a Low on this line when the PMMU detects a page fault, access violation, or write warning.

Table 4. Status Lines

ST_3-ST_0	Definition
0 0 0 0	Internal operation
0 0 0 1	Memory refresh
0 0 1 0	I/O reference
0 0 1 1	Special I/O reference (for example, to a PMMU)
0 1 0 0	Translation trap acknowledge
0 1 0 1	Nonmaskable interrupt acknowledge
0 1 1 0	Nonvectored interrupt acknowledge
0 1 1 1	Vectored interrupt acknowledge
1 0 0 0	Data memory request
1 0 0 1	Stack memory request
1 0 1 0	Data memory request (External Processing Architecture)
1 0 1 1	Stack memory request (External Processing Architecture)
1 1 0 0	Instruction space access
1 1 0 1	Instruction fetch, first word
1 1 1 0	External Processing Unit-CPU transfer
1 1 1 1	Bus lock, data memory request

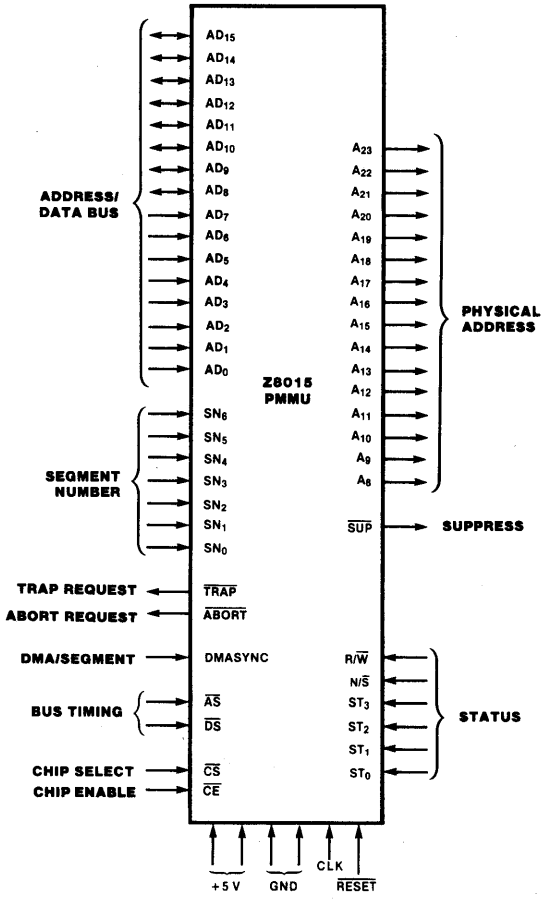


Figure 13. Pin Functions

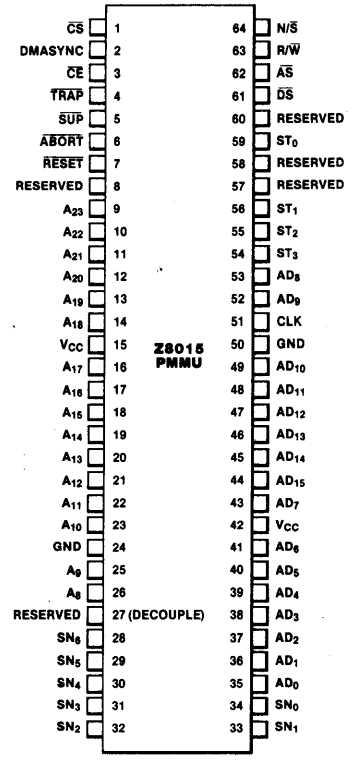


Figure 14. Pin Assignments

AC CHARACTERISTICS

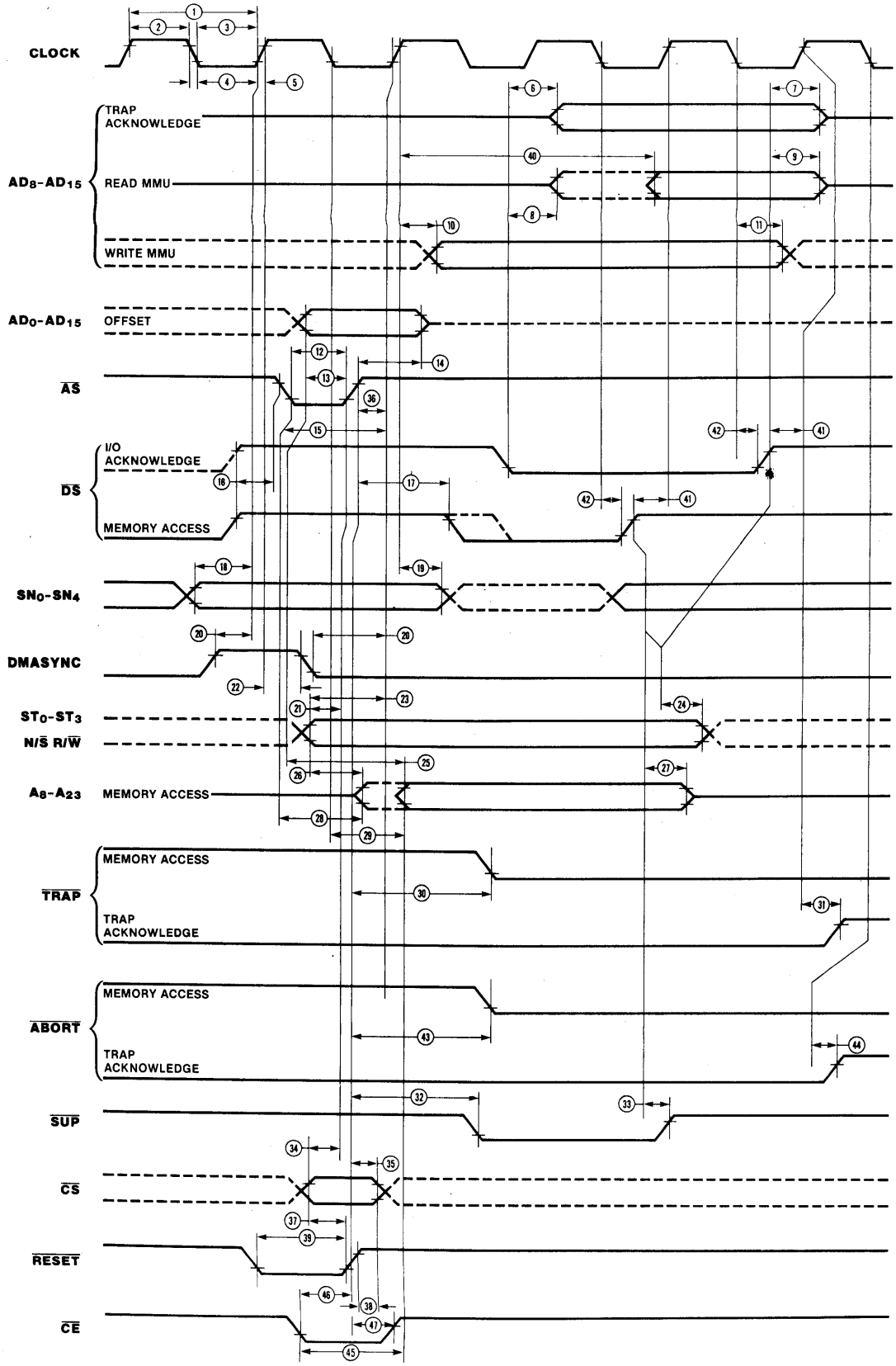
The following composite timing diagram shows the AC timing relationships of the PMMU's control, address, and data signals. The ac characteristics of these signal relationships are described in the AC Characteristics Table.

AC CHARACTERISTICS TABLE

No.	Symbol	Parameters	Min (4 MHz)	Max (4 MHz)	Notes
1	TcC	Clock Cycle Time	250		
2	TwCh	Clock Width (High)	105		
3	TwCl	Clock Width (Low)	105		
4	TfC	Clock Fall Time		20	
5	TrC	Clock Rise Time		20	
6	TdDSA(RDv)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		100	1
7	TdDSA(RDf)	$\overline{DS}$ ↑ (Acknowledge) to Read Data Float Delay	20	75	1
8	TdDSR(RDv)	$\overline{DS}$ ↓ (Read) to AD Output Driven Delay		100	1
9	TdDSR(RDf)	$\overline{DS}$ ↑ (Read) to Read Data Float Delay	20	75	1
10	TdC(WDv)	CLK ↑ to Write Data Valid Delay		160	
11	ThC(WDn)	CLK ↓ to Write Data Not Valid Hold Time	30		
12	TwAS	Address Strobe Width	60		
13	TsOFF(AS)	Offset Valid to $\overline{AS}$ ↑ Setup Time	45		
14	ThAS(OFFn)	$\overline{AS}$ ↑ to Offset Not Valid Hold Time	60		
15	TdAS(C)	$\overline{AS}$ ↓ to CLK ↑ Delay	110		
16	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	50		
18	TsSN(C)	SN Data Valid to CLK ↑ Setup Time	120		
19	ThC(SNn)	CLK ↑ to SN Data Not Valid Hold Time	0		
20	TdDMAS(C)	DMASYNC Valid to CLK ↑ Delay	120		
21	TdSTNR(AS)	Status (ST_0-ST_3, N/$\overline{S}$, R/$\overline{W}$) Valid to $\overline{AS}$ ↑ Delay	60		
22	TdC(DMA)	CLK ↑ to DMASYNC ↓ Delay	20		
23	TdST(C)	Status (ST_0-ST_3) Valid to CLK ↑ Setup Time	140		
24	TdDS(STn)	$\overline{DS}$ ↑ to Status Not Valid Delay	0		
25	TdOFF(Av)	Offset Valid to Address Output Valid Delay		200	1
26	TdST(Ad)	Status Valid to Address Output Driven Delay		180	1
27	TdDS(Af)	$\overline{DS}$ ↑ to Address Output Float Delay	30	160	1
28	TdAS(Ad)	$\overline{AS}$ ↓ to Address Output Driven Delay		170	1
29	TdC(Av)	CLK ↓ to Address Output Valid Delay		155	1
30	TdAS(TRAP)	$\overline{AS}$ ↑ to $\overline{TRAP}$ ↓ Delay		110	1,2
31	TdC(TRAP)	CLK ↑ to $\overline{TRAP}$ ↑ Delay		300	1,2
32	TdAS(SUP)	$\overline{AS}$ ↑ to $\overline{SUP}$ ↓ Delay		115	1,2
33	TdDS(SUP)	$\overline{DS}$ ↑ to $\overline{SUP}$ ↑ Delay	30	155	1,2
34	TsCS(AS)	Chip Select Input Valid to $\overline{AS}$ ↑ Setup Time	10		
35	ThAS(CSn)	$\overline{AS}$ ↑ to Chip Select Input Not Valid Hold Time	80		
36	TdAS(C)	$\overline{AS}$ ↑ to CLK ↑ Delay	0		
37	TsCS(RST)	Chip Select Input Valid to $\overline{RESET}$ ↑ Setup Time	150		
38	ThRST(CSn)	$\overline{RESET}$ ↑ to Chip Select Input Not Valid Hold Time	0		
39	TwRSTI	$\overline{RESET}$ Width (Low)	2TcC		
40	TdC(RDv)	CLK ↑ to Read Data Valid Delay		460	
41	TdDS(C)	$\overline{DS}$ ↑ to CLK ↑ Delay	30		
42	TdC(DS)	CLK ↓ to $\overline{DS}$ ↑ Delay	0	110	
43	TdAS(ABORT)	$\overline{AS}$ ↑ to $\overline{ABORT}$ ↓ Delay		110	
44	TdC(ABORT)	CLK ↓ to $\overline{ABORT}$ ↑ Delay	30	155	
45	TdCE(Av)	$\overline{CE}$ ↓ to Address Output Valid Delay		235	
46	TsCE(AS)	$\overline{CE}$ ↓ to $\overline{AS}$ ↑ Setup Time	0		
47	ThAS(CEn)	$\overline{AS}$ ↑ to Chip Enable Input Not Valid Hold Time	60		

NOTES:
1. 50 pF load.
2. 2.2K pull-up.
3. All times given in nanoseconds (ns).

AC TIMING DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0 °C to +70 °C

Storage Temperature . . . . . -65 °C to +150 °C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

The following dc characteristics apply for the given standard test conditions unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- $0\,°C \leq T_A \leq +70\,°C$

The type of test circuit used is as follows:

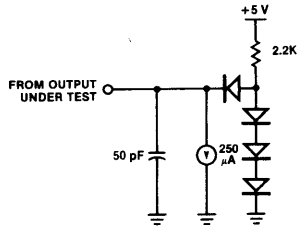


Figure 15. Test Load Circuit

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	$V_{CC}-0.4$	$V_{CC}+0.3$	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		300	mA	

ORDERING INFORMATION

Product Number	Package	Temperature Range	Speed	Description
Z8015	Ceramic	0 °C to +70 °C	4 MHz	Paged Memory Management Unit (Z-PMMU), 64-pin

The Z8016™ Z-DTC Direct Memory Access Transfer Controller

Zilog

Product Specification

September 1983

FEATURES

- Memory-to-peripheral transfers up to 2.66M bytes per second at 4 MHz.
- Memory-to-memory transfers up to 1.33M bytes per second at 4 MHz.
- Two fully independent, multi-function channels.
- Masked data pattern matching for Search and Transfer-and-Search operations.
- Funneling option that permits mixing of byte and word data during transfer operations.
- Can operate in logical address space with Zilog Memory Management Units, providing an 8M byte logical addressing range and 16M byte physical addressing range.
- Programmable chaining operation provides automatic loading of control parameters from memory into each channel.
- Software- or hardware-controlled Wait state insertion.
- Z-BUS™ daisy-chain interrupt hierarchy and bus-request structure.

GENERAL DESCRIPTION

The Z8016 DMA Transfer Controller (DTC) is a high performance data transfer device designed to match the power and addressing capability of the Z8000 CPUs. In addition to providing block data transfer capability between memory and peripherals, each of the two DTC channels can perform peripheral-to-peripheral and memory-to-memory transfers. A special Search mode of operation compares data read from memory or peripherals with the contents of a pattern register. A search can be performed concurrently with transfers or as an operation in itself.

In all operations (Search, Transfer, and Transfer-and-Search), the DTC can operate in either Flowthrough or Flyby transfer mode. In the Flowthrough mode, data is stored temporarily within the DTC on its way from source to destination. In this mode transfers can be made between a word-oriented memory and a byte-oriented peripheral through the bidirectional byte/word funneling option. In Flyby mode, data is transferred in a single step (from source to destination), thus providing twice the throughput.

The Z8016 DTC takes full advantage of the Z8000 memory management scheme by interfacing directly to the Z8010 Memory Management Unit (MMU) or the Z8015 Paged Memory Management Unit (PMMU). In this configuration, 8M bytes of logical address range are provided for each CPU address space. Alternatively, the Z8016 DTC can operate independently of an MMU, directly addressing up to 16M bytes of physical address space.

In addition to providing a hardware $\overline{\text{WAIT}}$ input to accommodate different memory or peripheral speeds, the Z8016 DTC allows the user to program the automatic insertion of either zero, one, two, or four Wait states for either source or destination addresses. Alternatively, the $\overline{\text{WAIT}}$ input pin function can be disabled and these software-programmed Wait states used exclusively.

The Z8016 DTC minimizes CPU involvement by allowing each channel to load its control registers from memory automatically when a DMA operation is complete. By loading the address of the next block of control parameters as part of this operation, command chaining is accomplished. The only action required of the CPU is to load the address of the control parameter table into the channel's Chain Address register and then issue a Start Chain command.

In some DMA applications, data is transferred continuously between the same two locations. To service these repetitive DMA operations, base registers are provided on each channel to reinitialize the current source and destination address registers. This re-initialization eliminates the need for reloading registers from memory tables.

The Z8016 DTC is directly Z-BUS compatible, and operates within the Z8000 daisy-chain vectored-priority interrupt scheme. The Demand Interleave operation allows the DTC to surrender the bus to the external system, or to alternate between internal channels. This capability allows for parallel operations between dual channels or between a DTC channel and the CPU.

The DTC can be used to provide a central DMA function for the CPU or to provide dispersed DMA operations in conjunction with a wide variety of Z8000 Family peripheral controllers.

The Z8016 DTC is packaged in a 48-pin DIP and uses a single +5 V power supply.

The Z8016 DTC pin functions and assignments are shown in Figures 1 and 2, respectively.

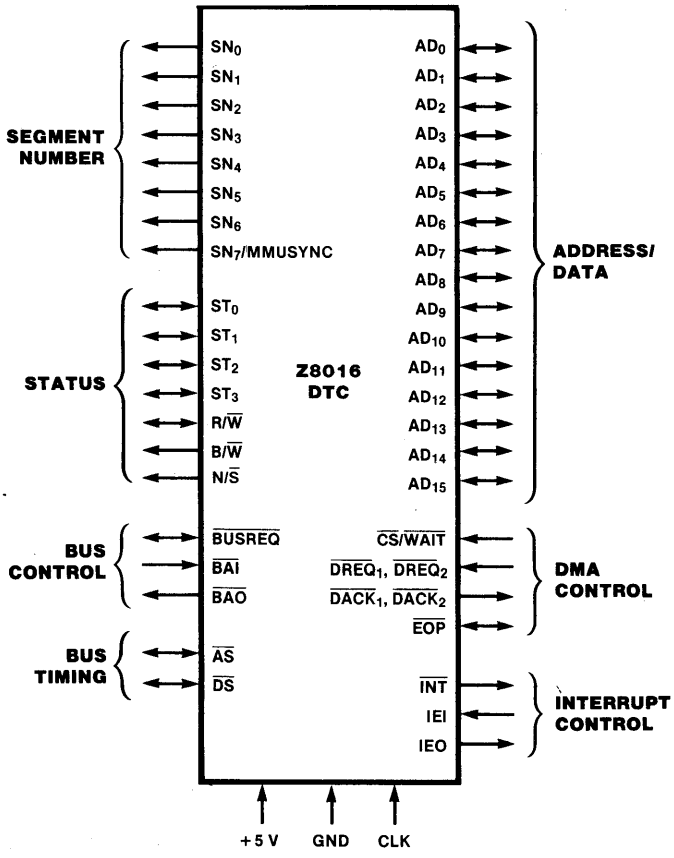


Figure 1. Pin Functions

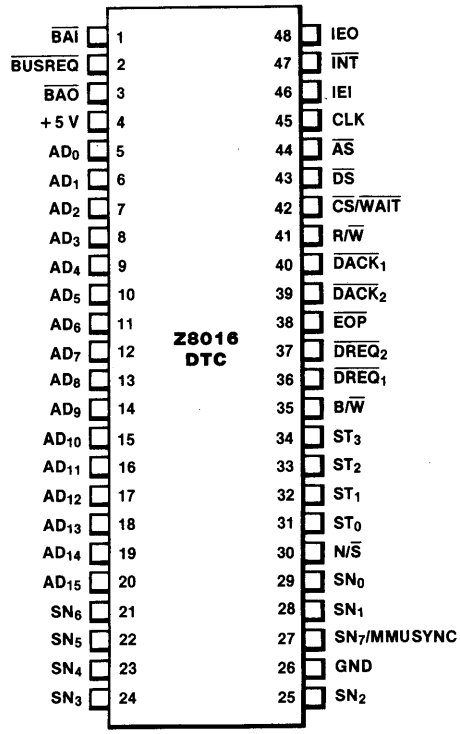


Figure 2. Pin Assignments

SIGNAL DESCRIPTIONS

AD_0-AD_{15}. *Address/Data Bus* (bidirectional, active High, 3-state) pins 5-20. These multiplexed Address/Data lines are used for all I/O and memory transactions.

$\overline{AS}$. *Address Strobe* (bidirectional, active Low, 3-state) pin 44. When the DTC is bus master the rising edge of $\overline{AS}$ (while $\overline{DS}$ is High) indicates that addresses are valid. When the DTC is not bus master, the address lines are sampled on the rising edge of $\overline{AS}$. There are no timing requirements between $\overline{AS}$ as an input and the DTC clock, because the Z-BUS does not use a bused clock. If $\overline{AS}$ and $\overline{DS}$ are simultaneously Low, the DTC will be reset.

$\overline{BAI}$. *Bus Acknowledge In* (input, active Low) pin 1. Signals that the bus has been released for DTC control. In multiple-DTC configurations, the $\overline{BAI}$ pin of the highest-priority DTC is normally connected to the Bus Acknowledge pin of the CPU. Each lower-priority DTC has its $\overline{BAI}$ connected to the $\overline{BAO}$ of the next higher-priority DTC.

$\overline{BAO}$. *Bus Acknowledge Out* (output, active Low) pin 3. In a multiple-DMA configuration, this pin signals that no higher-priority DTC has requested the bus. $\overline{BAI}$ and $\overline{BAO}$ form a daisy chain for multiple-DTC priority resolution.

BUSREQ. *Bus Request* (bidirectional, active Low, open-drain) pin 2. BUSREQ is used by the DTC to obtain control of the bus from the CPU. Before driving BUSREQ active, the DTC samples this line to ensure that another request is not already being made by another device. Since the DTC internally synchronizes the sampled BUSREQ signal, transitions on BUSREQ can be asynchronous with respect to the DTC clock.

B/W. *Byte/Word* (output, 3-state) pin 35. This output indicates the type of data transferred on the Address/Data (A/D) bus. A High on this line indicates a byte (8-bit) transfer and a Low indicates a word (16-bit) transfer. This signal is activated when AS goes Low and remains valid for the duration of the transaction.

CLK. *DTC Clock* (input) pin 45. The Clock signal controls internal operations and the rates of data transfer. It is usually derived from a master system clock or an associated CPU clock. When the DTC is used with an MMU, both must be driven from the same clock signal. While many DTC input signals are asynchronous, transitions for other signals (such as WAIT inputs) must meet setup and hold requirements relative to the DTC clock. (See the timing diagrams for details.)

CS/WAIT. *Chip Select/Wait* (input, active Low) pin 42. When the DTC is not in control of the system bus, this pin serves as a Chip Select (CS) input. A CPU or other external device uses CS to activate the DTC for reading and writing the DTC's internal registers. (CS can be held Low for multiple transfers to and from the DTC, provided that AS and DS are enabled for each transfer.) There are no timing requirements between the CS input and the DTC clock; the CS input timing requirements are only defined relative to AS.

When the DTC is in control of the system bus, this pin serves as the WAIT input. Slow memories and peripheral devices can use WAIT to extend DS during bus transfers. Unlike the CS input, transitions on the WAIT input must meet certain timing requirements relative to the DTC clock (see the Active State timing diagram for details). The WAIT function can be disabled using a control bit in the Master Mode register, in which case this input is treated as a Chip Select only and is ignored when the DTC is in control of the system bus.

$\overline{DACK}_1$, $\overline{DACK}_2$. *DMA Acknowledge* (output, active Low) pins 39 and 40. There is one DMA Acknowledge line associated with each channel. The DACK lines are programmed in the Channel Mode register to be pulsed, held active, or held inactive during DMA transfers. During Flyby operations the DACK line is used for two purposes. It selects the peripheral involved in the transfer, and it provides timing information on when to access the bus. During flowthrough operations the DACK line can be programmed to be active or inactive during a DMA transfer. DACK is not output during chaining operations.

$\overline{DREQ}_1$, $\overline{DREQ}_2$. *DMA Request* (input, active Low) pins 36 and 37. There is a DMA Request line associated with each channel. These lines can make transitions independent of the DTC clock. They are used by external logic to initiate and control DMA operations performed by the DTC.

DS. *Data Strobe* (bidirectional, active Low, 3-state) pin 43. A Low on this signal while AS is High indicates that the A/D bus is being used to transfer data. When the CPU is bus master and is transferring information to or from the DTC, DS is a timing input used by the DTC to move data to or from the A/D bus.

EOP. *End of Process* (bidirectional, active Low, open-drain, asynchronous) pin 38. This line is output when a Terminal Count (TC) or Match Condition (MC) termination occurs (see Termination section). An external source can terminate a DMA operation in progress by driving EOP Low. EOP always applies to the active channel; if no channel is active, EOP is ignored. The Suppress output of the MMU can be connected to EOP to terminate DMA accesses that violate the MMU protection settings. To provide full access protection, an external EOP is accepted even during chaining.

IEI. *Interrupt Enable In* (input, active High) pin 46. IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher-priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High) pin 48. IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting DTC. IEO is connected to the next lower-priority device's IEI input and thus inhibits interrupts from lower-priority devices.

INT. *Interrupt Request* (output, open-drain, active Low) pin 47. This signal is pulled Low when the DTC requests an interrupt.

N/S. *Normal/System* (output, 3-state) pin 30. The N/S signal is activated when the DTC is bus master. The N/S signal indicates which memory space is being accessed by going High for normal memory and Low for system memory.

R/W. *Read/Write* (bidirectional, 3-state, Low = write) pin 41. When the DTC is not bus master, R/W is a status input used to indicate whether data is being read from (High) or written to (Low) the DTC. When the DTC is bus master, R/W is an output used to indicate whether the DTC is reading or writing the addressed location. During Flyby DMA operations, the "Flyby peripheral" (Figure 3) inverts the R/W signal to determine whether it must read or write.

SN_0–SN_6. *Segment Number* (output, 3-state) pins 21-25 and 28-29. In logical address configuration, these lines provide the segment number field of a 23-bit segmented address. The SN_0–SN_6 I/O address information can be used to increase the DTC's logical I/O address space beyond that of the CPU. In physical address configuration, these lines provide bits 23 through 17 of a 24-bit linear address. The 24th bit (MSB) is output on SN_7/MMU Sync.

SN_7 or MMU Sync. *Segment Number 7 or MMU Sync* (output, 3-state) pin 27. In a logical address space configuration (with MMU), this line outputs an active High pulse prior to each machine cycle. The MMU uses this signal to synchronize access to its translation table and to differentiate between CPU and DTC control. The MMU ignores MMUSYNC if the status lines (ST_0–ST_3) indicate that an I/O transaction is being performed. This output is Low when the DTC is not bus master and the MM1 bit in the Master Mode register is set.

In a physical address space configuration (without MMU), this line outputs SN_7, which becomes the 24th address bit in a linear address space. The 24-bit linear address configuration allows the DTC to access 16M bytes of memory. This pin floats to the high impedance state when the DTC is not bus master and the MM1 bit is cleared.

ST_0–ST_3. *Status* (bidirectional, 3-state) pins 31-34. When the DTC is bus master, these lines are outputs indicating the type of memory or I/O transaction being performed. When the DTC is not bus master, the status lines are inputs used to detect Interrupt and Segment Trap Acknowledge cycles (Table 1).

Table 1. Status Codes

ST_3	ST_2	ST_1	ST_0	Transaction/Operation	Status Code Generated/Decoded
0	0	0	0	Internal Operation	
0	0	0	1	Memory Refresh	
0	0	1	0	I/O Transaction	Generated
0	0	1	1	Special I/O Transaction	Generated
0	1	0	0	Segment Trap Acknowledge	Decoded
0	1	0	1	Nonmaskable Interrupt Acknowledge	Decoded
0	1	1	0	Nonvectored Interrupt Acknowledge	Decoded
0	1	1	1	Vectored Interrupt Acknowledge	Decoded
1	0	0	0	Memory Transaction for Data/DTC Chaining	Generated
1	0	0	1	Memory Transaction for Stack	Generated
1	0	1	0	Reserved	
1	0	1	1	Reserved	
1	1	0	0	Memory Transaction for Program Fetch (Subsequent Word)	Generated
1	1	0	1	Memory Transaction for Program Fetch (First Word)	
1	1	1	0	Reserved	
1	1	1	1	Reserved	

FUNCTIONAL DESCRIPTION

Channel Initialization

The Z8016 DTC operates with a minimum of interaction with the host CPU. Each channel's operation is determined by the settings of its own set of control registers. Each channel is initialized when the DTC loads its control parameters from memory into its control registers during the chaining operation. To initiate the chaining operation, the CPU is required to program the Master Mode register and each channel's Chain Address register. Then each channel's control registers are automatically loaded by the DTC with control parameters stored in a chain control table in memory, located at the address pointed to by that channel's Chain Address register. Once the channel registers are loaded, the DTC is ready to perform DMA operations.

Initiating DMA Operations. DMA operations can be initiated in three ways:

- Software Request. The CPU can issue Software Request commands to start DMA operations on a specific channel. This channel must then request control of the bus and perform transfers.
- Hardware Request. DMA operations can be started by forcing a channel's $\overline{DREQ}$ input Low, as described in the Transfer Modes section.
- Starting After Chaining. If the Software Request bit of the Channel Mode register is loaded with a 1 during chaining, the channel will perform the programmed DMA operation at the end of chaining. If the channel is

programmed for Single Operation or Demand mode, it will perform the operation immediately. The channel will give up the bus after chaining and before the operation if the CPU Interleave bit in the Master Mode register is set. Note that once a channel starts a chaining operation by fetching a reload word, it retains bus control at least until all of the registers specified in the reload word have been loaded from memory.

Transfers

The Z8016 DTC uses three basic types of operation: Transfer, Search, and Transfer-and-Search.

During a Transfer operation, the DTC obtains control of the system A/D bus from the CPU. Data is read from one addressable port (source) and is written to another addressable port (destination) in words or bytes. This applies to both Flyby and Flowthrough transfers.

Flyby transfers use a single addressing/transfer cycle, in which data is transferred directly from the source to the destination with no intermediate storage (Figure 3). This method of transfer provides higher throughput than Flowthrough transfers but cannot be used for memory-to-memory transfer.

Flowthrough transfers are used for all combinations of addressable memory and I/O spaces. These transfers use independent double Addressing/Transfer cycles, in which data is stored temporarily in the DTC while being transferred from source to destination (Figure 4). Flowthrough transfers can use the funneling option, which allows mixing of data sizes between source and destination. For example, a byte-oriented peripheral can conveniently supply data to a word-oriented memory. This option requires no added circuitry for either memory or peripherals.

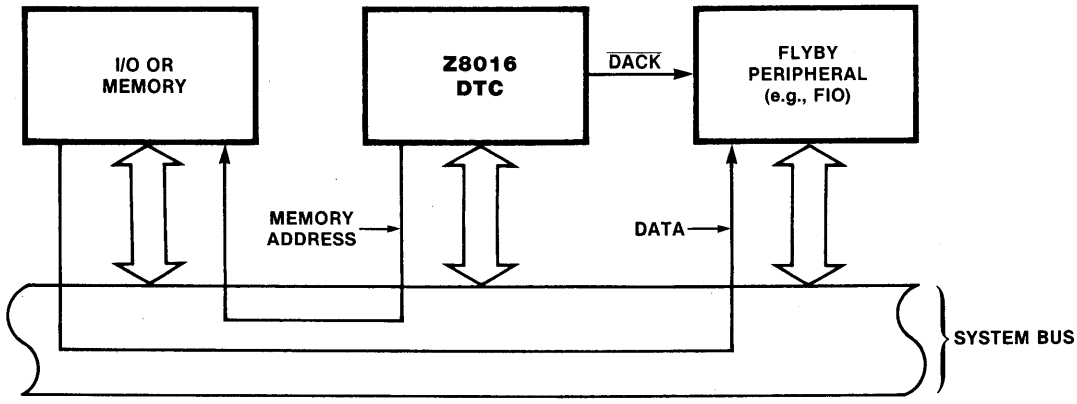


Figure 3. Configuration of a Flyby Transaction

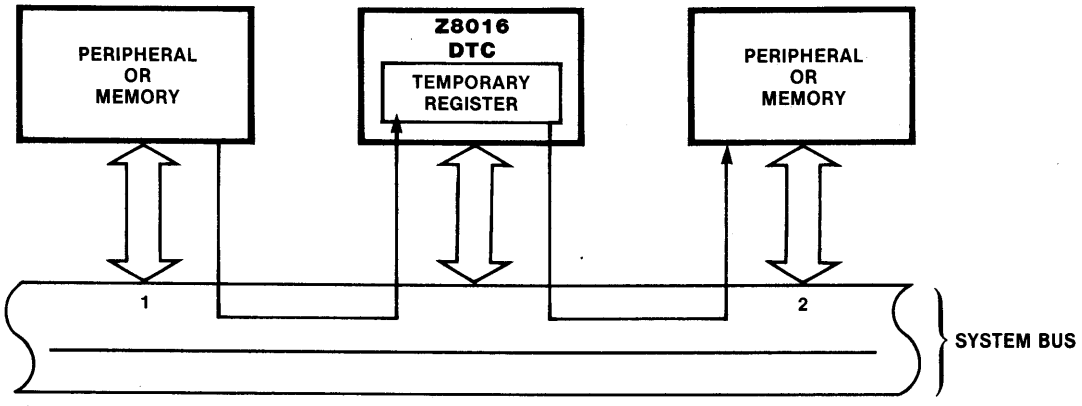


Figure 4. Configuration of a Flowthrough Transaction

During a Search operation, data is read from the source port and compared byte-by-byte with a pattern register containing a programmable match byte. The Search operation can be programmed to stop either when the read data matches (Stop-on-Match) or when it fails to match the masked pattern (Stop-on-No-Match). For word reads, the Channel Mode register can be used to select either 8- or 16-bit compares.

Transfer-and-Search operations combine the transfer and search functions to facilitate the transfer of variable-length data blocks. While data is being transferred between two ports, a simultaneous search is made for a bit-maskable byte match. Transfer-and-Search can be performed in either Flowthrough or Flyby mode. A Flyby Transfer-and-Search can be used to increase throughput for transfers between peripherals or between memory and a peripheral; it cannot be used for memory-to-memory transfers.

Transfer Modes. The Z8016 DTC operates in either of two transfer modes: Single or Demand. The Demand mode is further divided into the Demand Dedicated with Bus Hold, Demand Dedicated with Bus Release, and Demand Interleave modes.

The Single mode is used with peripherals that transfer single bytes or words at irregular intervals. Each Software Request command causes the channel to perform a single DMA operation and each application of a High-to-Low transition on the $\overline{\text{DREQ}}$ input also initiates a DMA operation. Each time a Single mode DMA operation ends, the channel relinquishes the bus unless a new transition has occurred on $\overline{\text{DREQ}}$.

In the Demand mode, when the $\overline{\text{DREQ}}$ input is active, transfer cycles are executed repeatedly until the transfer is completed. In the Demand Dedicated with Bus Hold mode, the active channel retains control of the bus until the transfer is complete, even after the $\overline{\text{DREQ}}$ input has gone inactive. In the Demand Dedicated with Bus Release mode, the active channel releases control of the bus when the $\overline{\text{DREQ}}$ input goes inactive. When the $\overline{\text{DREQ}}$ input becomes active again, control of the bus is re-acquired and the transfer operation continues.

The Demand Interleave mode has two options, programmable in the Master Mode register bit MM2. If MM2 is set, the DTC relinquishes and re-requests bus control after every DMA operation.

This permits the CPU and other devices to gain bus control. If both channels receive active $\overline{\text{DREQ}}$ inputs, each channel relinquishes control to the CPU after each operation. In the second option (MM2 is 0), control can pass from one channel to the other without requiring the DTC to release bus control. If both channels receive active $\overline{\text{DREQ}}$ inputs, control alternates between channels and the DTC retains bus control until all channel operations are complete.

Wait States. The Z8016 DTC can insert Wait cycles into the DMA Transaction cycle under hardware or software control. The $\overline{\text{CS}}/\overline{\text{WAIT}}$ input can be multiplexed to function as a Chip Select for the DTC when it does not have control of the bus, and as a $\overline{\text{WAIT}}$ input when the DTC is the bus controller. Multiplexing $\overline{\text{CS}}$ and $\overline{\text{WAIT}}$ requires external logic, but the DTC can be programmed to insert Wait states automatically without external logic when accessing either I/O or memory addresses. Either zero, one, two, or four Wait states can be added. Wait states can be programmed separately for the Current Address registers and for the Chain Address register. Programmable Wait cycle insertion allows memories and peripherals of different speeds to be associated with I/O and memory addresses.

Interrupts. On the Z8016 DTC, each channel is an interrupt source and has its own vector register for identifying the source of the interrupt during a CPU/DTC Interrupt Acknowledge transaction. An interrupt can result from a Match Condition (MC), End-Of-Process (EOP), or Terminal Count (TC) on either channel. The user selects the action to be performed by setting bits in the Channel Mode register.

Three bits in each channel's Status register control interrupts. These are the Channel Interrupt Enable (CIE) bit, the Interrupt Pending (IP) bit, and the Interrupt Under Service (IUS) bit.

Devices connected to any of the CPU's three interrupt inputs resolve priority conflicts with an interrupt daisy chain, as shown in Figure 5. The daisy chain has two functions. During an Interrupt Acknowledge transaction, it determines which interrupt source is being acknowledged. At all other times, it determines which interrupt sources can initiate an interrupt request.

The Z8016 DTC has an interrupt queuing capability, which includes a two-deep interrupt queue on each channel. This allows the DTC to continue normal operation between the time an interrupt is issued and the time the Interrupt Acknowledge is received.

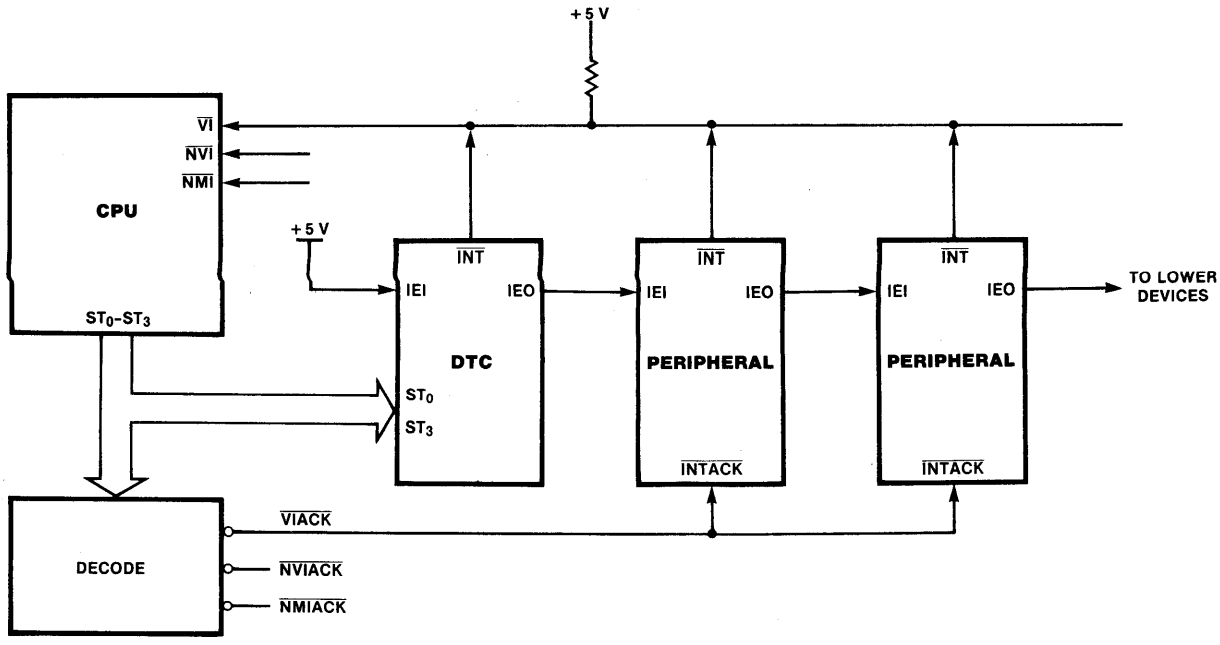


Figure 5. Interrupt Daisy Chain

Termination

DMA operations can end in one of the following three ways:

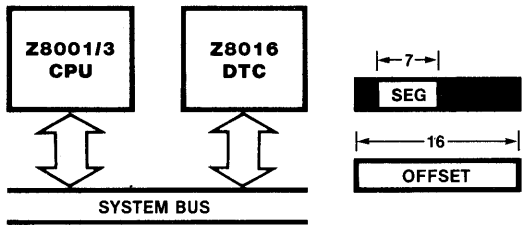
- A Terminal Count (TC) termination occurs when a channel's Current Operation Count register goes to 0.
- An End-of-Process (EOP) termination occurs when the DTC'S EOP pin is driven Low by external logic.
- A Match Condition (MC) termination occurs when data being Searched or Transferred-and-Searched meets the match condition programmed in the Channel Mode register.

MEMORY MANAGEMENT

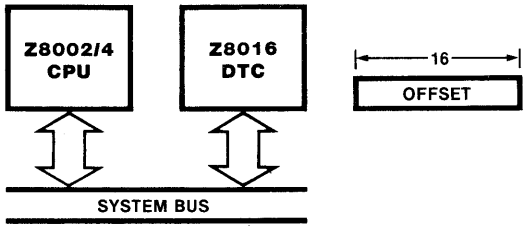
The DTC can be configured to operate in physical address space or logical address space. When the DTC is operated in logical address space, the segment and offset portions of the address registers combine to form 23-bit logical addresses. In conjunction with a CPU, DMA operations can be handled through the Z8010 MMU or the Z8015 PMMU. MMUs offer dynamic segment relocation, segment protection, and other memory management features.

In the physical address space configuration, the segment and offset portions of the DTC's address registers are combined with the SN_7 output to form a single 24-bit linear address. The extended I/O addressing capability of the DTC can be used to increase the DTC's physical I/O address space beyond that of the CPU. Figure 6 illustrates various DTC configuration options with the Z8000 CPUs and MMUs.

DTC WITH Z8001/3 (SEGMENTED) CPU



DTC WITH Z8002/4 (NONSEGMENTED) CPU



DTC WITH Z8001/3 (SEGMENTED) CPU AND MMU

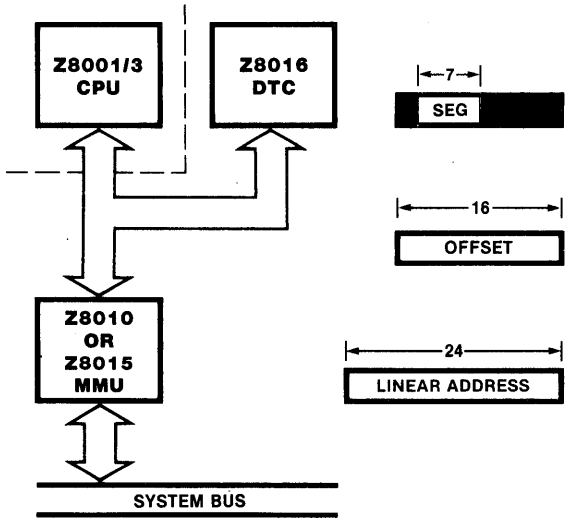


Figure 6. DTC Configurations

INTERNAL STRUCTURE

The internal structure of the Z8016 DTC includes driver and receiver circuitry for interfacing with Zilog's Z-BUS. The DTC's internal bus interfaces with the Z-BUS and services all internal logic and registers, as illustrated in the DTC block diagram (Figure 7).

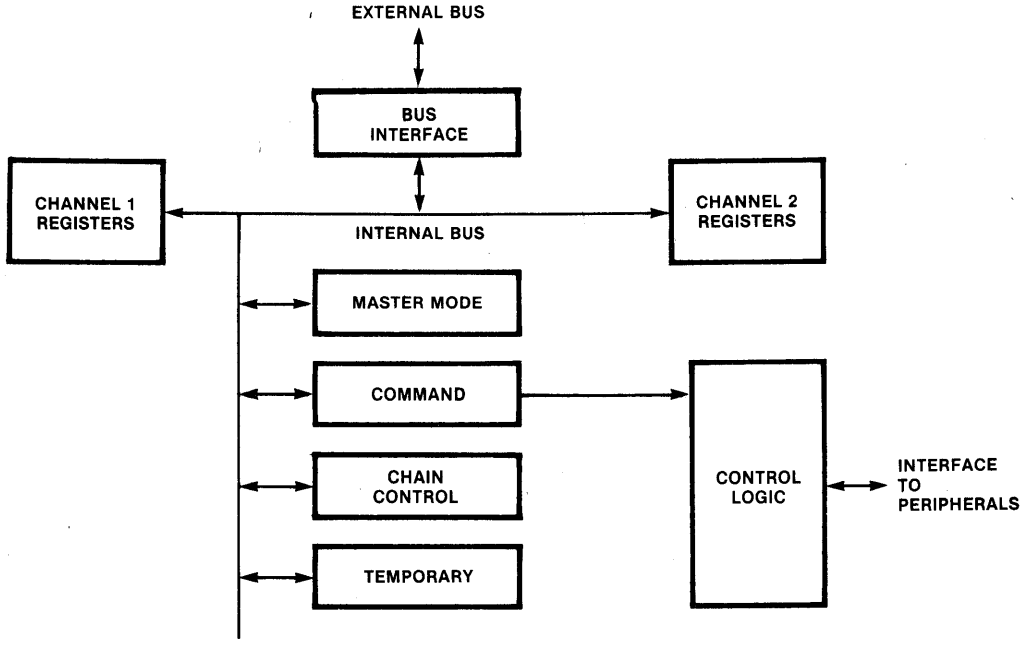


Figure 7. DTC Block Diagram

REGISTER DESCRIPTION

The DTC contains chip-level control registers as well as channel-level registers that are duplicated for each channel. Registers on the DTC that can be read by the CPU are either fast- or slow-readable. CPU I/O instructions can read fast-readable registers without Wait states. Slow-readable registers can be read by the CPU only if Wait states are inserted. This requires external logic to generate and time the application of Low signals on the CPUs $\overline{WAIT}$ input if the slow-readable registers are to be read.

Control Registers

The four control registers direct the functioning of the DTC. (Figure 8.)

Master Mode Register. This register selects the way in which the DTC interfaces to the system. The following descriptions indicate how the individual bits in the Master Mode register are used. The Master Mode register is fast-readable.

Chip Enable (CE). The setting of this bit enables the DTC to request the bus, perform DMA operations and reload registers.

Logical/Physical Address Space (LPA). The setting of this bit determines how the system will view the segment and offset portions of the Current ARA and ARB registers. When LPA is set to 1 (Logical Address Space), the segment and offset portions of the Current ARA and ARB registers are treated as separate portions of the address. The 16-bit offset portion of the address will appear on pins AD_0–AD_{15} when $\overline{AS}$ is Low. The 7-bit segment number appears on pins SN_0–SN_6 for the duration of the transaction.

When this bit is set to 0 (Physical Address Space), the segment and offset portions of the Current ARA and ARB registers are treated as a single address and all eight segment bits in the register are used. Both the I/O and the memory addresses in Physical Memory Space are generated by loading the offset portion of the Current Address register onto the AD_0–AD_{15} bus and the segment portion of that register onto the SN_0–SN_7 bus. (In conjunction with the nonsegmented Z8000 CPUs, either Logical or Physical Address Space setting may be used.)

Wait Line Enable (WLE). This bit is set to enable sampling of the $\overline{CS}/\overline{WAIT}$ line during memory and I/O transactions.

Disable Lower Chain (DLC). This bit is set to inhibit all lower priority devices on the interrupt daisy chain. While DLC is 0, the DTC generates Low and High signals on the IEO output in response to IEI.

No Vector on Interrupt (NVI). This bit determines whether the DTC channel or a peripheral returns a vector during Interrupt Acknowledge cycles. While the bit is cleared, a channel receiving an Interrupt Acknowledge will drive the contents of its Interrupt Save register onto the A/D bus while $\overline{DS}$ is Low. While this bit is set, interrupts are serviced in an identical manner, but the A/D bus remains in a high impedance state throughout the Acknowledge cycle.

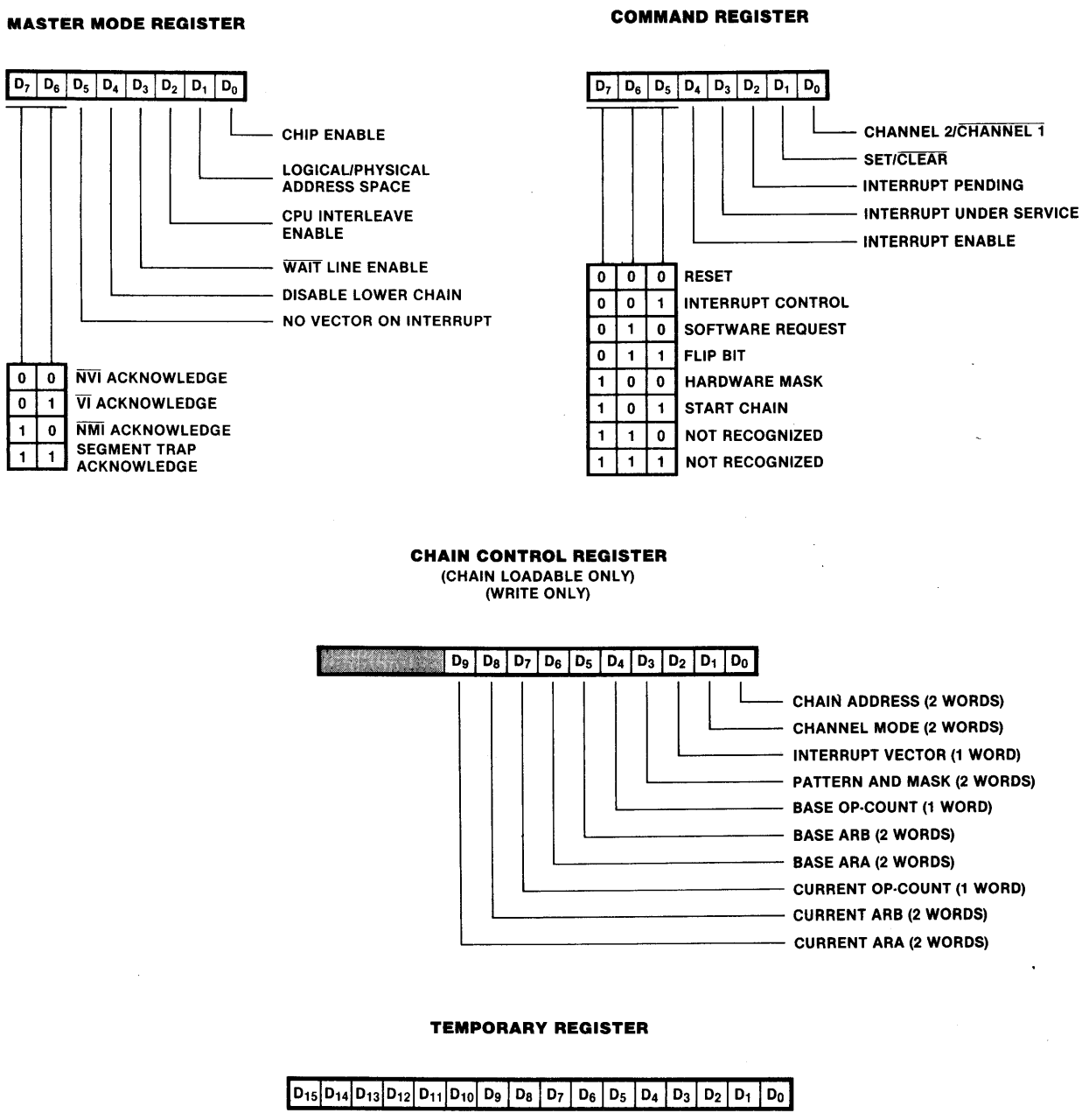


Figure 8. Control Registers

Interrupt Acknowledge Field (two bits). This field is used to select the type of Interrupt Acknowledge cycle the DTC is to respond to. The setting of this field must correspond to the IEI/IEO daisy chain on which the DTC is located. The DTC can respond to Nonmaskable Interrupt (NMI), Nonvectored Interrupt (NVI), or Segment Trap Acknowledge cycles.

CPU Interleave Enable. When this bit is set, interleaving of bus use between the CPU and the DTC is enabled.

Chain Control Register. This 16-bit register specifies which registers are to be loaded from memory during a chaining operation. The Chain Control register is loaded from the memory location pointed to by the Chain Address register. The Chain Control register is chain loadable only and cannot be accessed by the CPU.

Command Register. The Command register is an 8-bit write-only register written to by the host CPU to execute commands. The Command register is loaded from the data on AD_7–AD_0; the data on AD_{15}–AD_8 is disregarded.

Temporary Register. This 16-bit register is used to hold data during Flowthrough transfers, Search operations, and Transfer-and-Search operations. The Temporary register cannot be written or read by the CPU.

Channel-Level Registers

Each of the DTC's two channels has a complete set of channel-level registers. This set consists of both General-Purpose and Special-Purpose registers, as illustrated in Figure 9. The General-Purpose registers are commonly found on DMA devices and can be read or written by the CPU. The Special-Purpose registers provide additional features specific to the Z8016 DTC.

General-Purpose Registers. The General-Purpose register set on each channel consists of the Current Address registers A and B, the Base Address registers A and B, the Base and Current Operation Count registers, and the Channel Mode register (Figure 10).

Current and Base Address Registers A and B. The Current Address registers A and B are used to point to the source and destination for DMA operations. The contents of the Base Address registers A and B are transferred into the Current Address registers A and B at the end of a DMA operation if the user enables base-to-current reloading in the Completion field of the Channel Mode register. The base-to-current reload operation facilitates repetitive DMA operations without the multiple memory accesses required by chaining.

Each of the Base and Current Address registers A and B consist of two words. The first word contains a 7-bit Tag field and an 8-bit Segment Number field. The second word contains a 16-bit offset. The use of the Tag field is described below. The use of the Segment Number field depends upon the setting of the LPA bit in the Master Mode register. The Base and Current Address registers are fast-readable and can be loaded by chaining.

Programmable Wait Field. This field allows the insertion of zero, one, two, or four Wait states into memory or I/O accesses addressed by the offset and segment fields.

Address Control Field. At the end of each iteration of a DMA operation, the address can be incremented, decremented, or left unchanged. Memory addresses are changed by one if the address points to a byte operand or by two if the address points to a word operand.

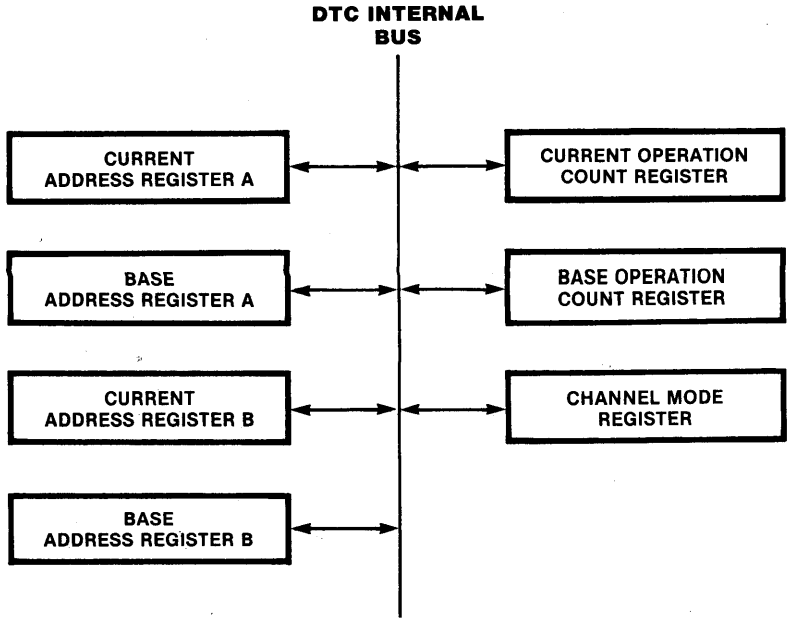
Address Reference Field. This portion of the Tag field is used to select whether the address pertains to memory space or I/O space. The N/$\overline{S}$ output line is always Low (indicating System) for I/O space but can be either High (Normal) or Low (System) for memory space.

Current and Base Operation Count Registers. The 16-bit Current Operation Count register specifies the number of words or bytes to be transferred, searched, or transferred-and-searched. For word-to-word operations and byte-word funneling, this register must be programmed with the number of words to be transferred or searched.

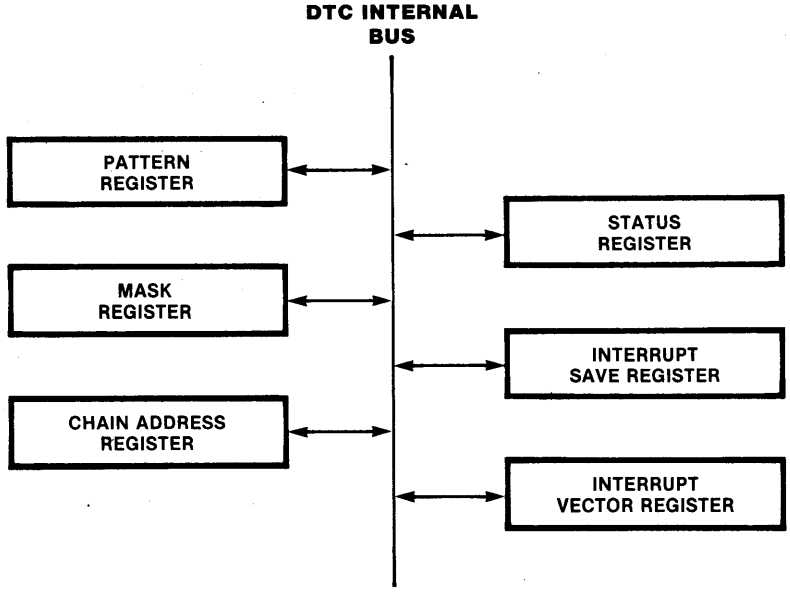
The Base Operation Count register reinitializes the current source and destination in the Current Operation Count register. Each time data is transferred or searched, the Current Operation Count register is decremented by one. Once all of the data is transferred or searched, the Current Operation Count register will contain zero. If the transfer on search stops before the Current Operation Count register reaches zero, the contents of the register indicate the number of bytes or words remaining to be transferred or searched. This allows a channel to be restarted from where it left off without requiring reloading of the Current Operation Count register. The Current and Base Operation Count registers are slow-readable and can be loaded by chaining.

Channel Mode Register. This register selects the type of DMA operation the channel is to perform, how the operation is to be executed, and what action is to be taken when the operation finishes. The Channel Mode register is slow-readable and can be loaded by chaining.

Data Operation and Transfer Type Field. These fields are used to select the type of operation the channel is to perform along with the operand size. The specific codes are listed in Tables 2 and 3. The Flip bit is used to select which of the Current Address Registers A (ARA), or B (ARB), points to the source and which points to the destination address.



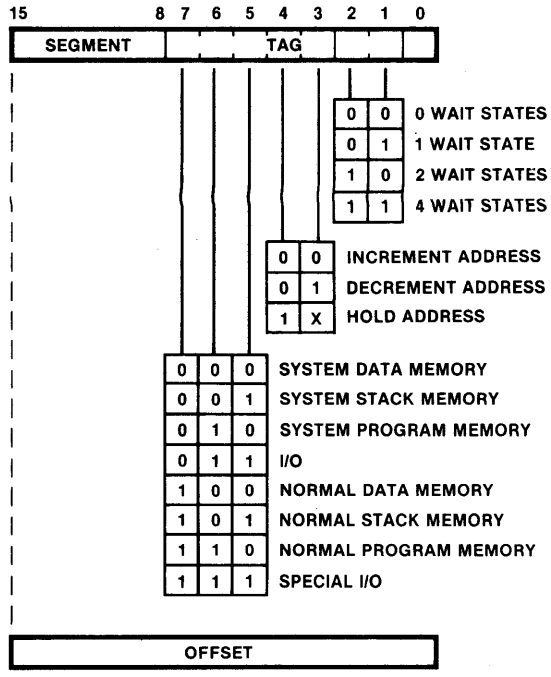
GENERAL-PURPOSE CHANNEL REGISTERS



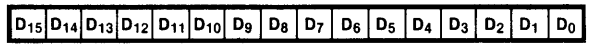
SPECIAL-PURPOSE CHANNEL REGISTERS

Figure 9. Channel-Level Registers

BASE AND CURRENT ADDRESS REGISTERS A AND B



BASE AND CURRENT OPERATION COUNT REGISTERS



CHANNEL MODE REGISTER

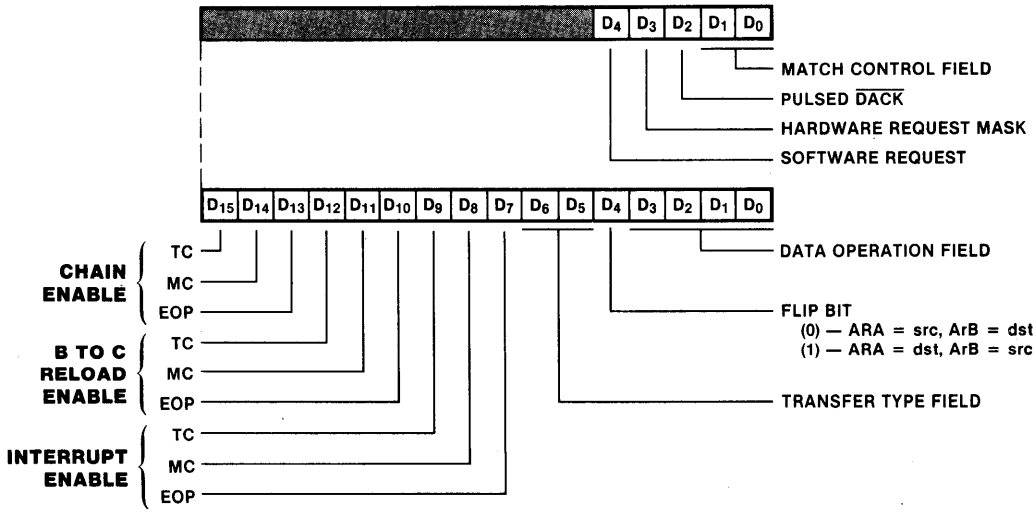


Figure 10. General-Purpose Channel Registers

Table 2. Data Operation Field

Code/Operation	Operand Size ARA	Operand Size ARB	Transaction Type
	Transfer		
0001	Byte	Byte	Flowthrough
100X	Byte	Word	Flowthrough
0000	Word	Word	Flowthrough
0011	Byte	Byte	Flyby
0010	Word	Word	Flyby
	Transfer-and-Search		
0101	Byte	Byte	Flowthrough
110X	Byte	Word	Flowthrough
0100	Word	Word	Flowthrough
0111	Byte	Byte	Flyby
0110	Word	Word	Flyby
	Search		
1111	Byte	Byte	N/A
1110	Word	Word	N/A
101X	Illegal		

Completion Field. This field is used to program the action taken by the channel at the end of a DMA operation. When a DMA operation ends, the channel can perform any combination of the following options:

- Interrupt the CPU (Interrupt Enable field)
- Base-to-Current reload (B to C Reload field)
- Chain reload the next DMA operation (Chain Enable field)

The options are performed according to the bits set in the Interrupt Enable, B to C Reload, and Chain Enable fields for each type of termination that occurs; the NAC bit in the Status register is automatically set on completion of a DMA operation.

Match Control Field. This 2-bit field determines whether matches use an 8-bit or 16-bit pattern and whether the channel is to Stop-On-Match or Stop-On-No-Match. The specific codes for the Match Control field are listed in Table 3.

Table 3. Transfer Type Field and Match Control Field

Code	Transfer Type	Match Control
00	Single Transfer	Stop on No Match
01	Demand Dedicated/Bus Hold	Stop on No Match
10	Demand Dedicated/Bus Release	Stop on Word Match
11	Demand Interleave	Stop on Byte Match

Pulse $\overline{\text{DACK}}$ (PD). This bit determines when the $\overline{\text{DACK}}$ line is active. While cleared, the channel's $\overline{\text{DACK}}$ line is active whenever the channel is performing a DMA operation, regardless of the type of transaction. While the PD bit is set, the $\overline{\text{DACK}}$ pin is inactive during chaining, Flowthrough Transfers, Flowthrough Transfer-and-Searches, and Searches. $\overline{\text{DACK}}$ is pulsed active during Flyby Transfers and Flyby Transfer-and-Searches at the time necessary to strobe data into, or out of, the Flyby peripheral.

Hardware Request Mask (HRM). If this bit is set, a DMA operation can be started by applying a Low on the channel's $\overline{\text{DREQ}}$ input.

Software Request (SR). If this bit is set during chaining, the channel performs the programmed DMA operation at the end of the chaining operation.

Special Purpose Registers. The Special-Purpose registers on each channel are the Pattern and Mask registers, the Status register, the Interrupt Vector register, the Interrupt Save registers, and the Chain Address register (Figure 11).

Pattern and Mask Registers. These registers are used in Search and Transfer-and-Search operations. The Pattern register contains the pattern that the read data is compared to. The Mask register allows the user to exclude or mask selected Temporary register bits from comparison by setting the corresponding Mask register bit to 1. The Pattern and Mask registers are slow-readable and can be loaded by chaining.

Status Register. The Status register on each channel reports the status of that channel. The functions of the individual bits are indicated in the following field descriptions. The Status register is fast-readable.

Completion Status Field. Three bits indicate whether the DMA operation ended as a result of TC, MC, or EOP. The TC bit is set if the Operation Count (reaching zero) ends the DMA operation. The MC bit is set if a pattern match termination occurs. The EOP bit is set when an EOP termination ends a DMA transfer. The appropriate combination of the TC, MC, and EOP bits is set if multiple reasons exist for ending a DMA operation. The Match Condition High byte (MCH) and Match Condition Low byte (MCL) bits report the match states of the upper and lower comparator bytes of the last word transferred. The MCH and MCL bits are updated with each transfer.

These bits are set when the associated comparator bytes are matched, regardless of whether Stop-on-Match or Stop-on-no-Match is programmed.

Hardware Interface Status Field. The Hardware Request (HRQ) bit provides a means of monitoring the channel's $\overline{\text{DREQ}}$ input line. While $\overline{\text{DREQ}}$ is Low, the HRQ bit is set. While the Hardware Mask (HM) bit is set, the DTC is prevented from responding to a Low on the $\overline{\text{DREQ}}$ line. However, the HRQ bit always reports the status of $\overline{\text{DREQ}}$ regardless of the status of the HM bit.

DTC Status Field. This field reports the current channel status to the CPU. The "channel initialized and waiting for request" status is implicitly indicated if bits ST_{12} through ST_9 are clear.

Second Interrupt Pending (SIP). When a second interrupt is to be issued before the first interrupt is acknowledged, this bit is set and the channel relinquishes the bus until an Acknowledge occurs.

Waiting for Bus (WFB). This bit is set when the channel is waiting for bus control to perform a DMA operation.

No Auto-Reload or Chaining (NAC). This bit is set under the following conditions:

- A channel completes a DMA operation and neither Base-to-Current reloading nor auto-chaining is enabled.
- A channel is issued an $\overline{EOP}$ during chaining.
- A Reset is issued to the DTC.

Chaining Abort (CA). This bit is set when a channel is issued an $\overline{EOP}$ during chaining or a Reset is issued to the DTC. The Chain Abort (CA) bit holds the No Auto-Reload or Chaining (NAC) bit in the set state until the $\overline{EOP}$ bit is cleared. The CA bit is cleared when a new Chain Address Segment and Tag word or Offset word is loaded into the channel.

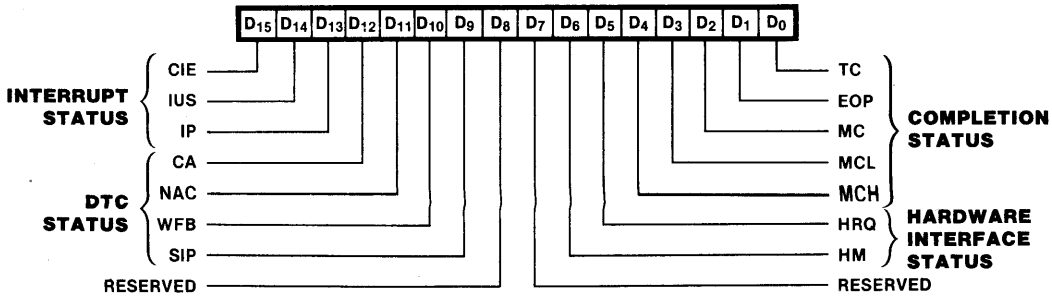
Interrupt Status Field. The Channel Interrupt Enable (CIE), Interrupt Pending (IP), and Interrupt Under Service (IUS) bits are used to control the way a channel generates an interrupt. An interrupt source with its IP bit set makes an interrupt request if all of the following conditions are met: Interrupts are enabled, (CIE bit = 1), there is no Interrupt Under Service (IUS bit = 0), no higher priority interrupt is being serviced, and no Interrupt Acknowledge transaction is in progress. When an interrupt source has an Interrupt Under Service (IUS = 1), all lower priority interrupt sources are prevented from requesting interrupts.

Interrupt Vector and Interrupt Save Registers. The 8-bit Interrupt Vector register contains the vector or identifier to be output during an Interrupt Acknowledge cycle. When an interrupt occurs, the contents of the Interrupt Vector register and bits ST_9–ST_{15} of the Status register are stored in the 16-bit Interrupt Save register. Because the vector and status are stored, a new vector can be loaded during chaining and a new DMA operation can be performed before an Interrupt Acknowledge cycle occurs. If another interrupt occurs on the channel before the first is acknowledged, further channel activity is suspended. When a clear IP command is issued, the status and vector for the second interrupt are loaded into the Interrupt Save register and channel operation resumes. The DTC can retain only two interrupts for each channel. The Interrupt Save register is fast-readable.

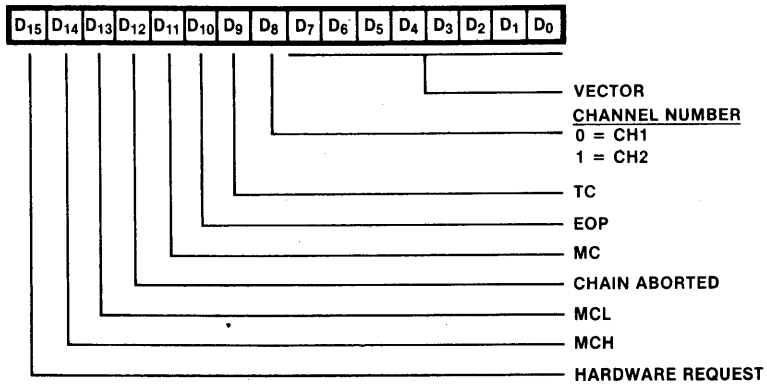
Chain Address Register. This register points to the chain control table in memory containing data to be loaded into the channel's registers. The Chain Address register consists of two words (Figure 11). The first word consists of a Segment and Tag field. The second word contains the 16-bit offset portion of the memory address. Bit 15 in the Segment field is ignored when the DTC is configured for logical address space (LPA = 1). The Tag field contains two bits used to designate the number of Wait states to be inserted during accesses to the Chain Control table. The Chain Address register is fast-readable and is loadable by chaining.

Table 4 provides a list of register addresses.

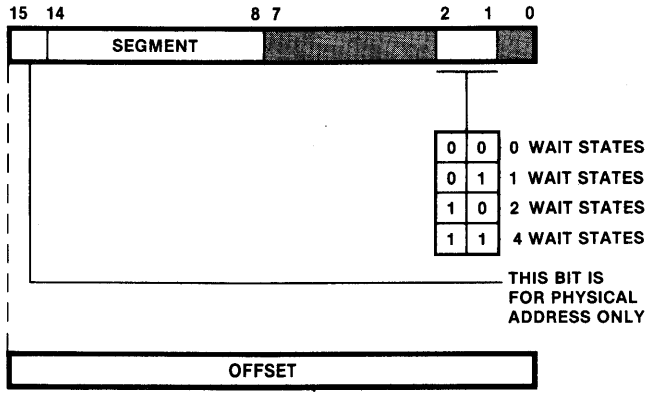
STATUS REGISTER



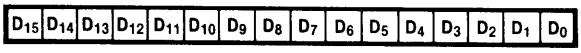
INTERRUPT SAVE REGISTER



CHAIN ADDRESS REGISTER



PATTERN AND MASK REGISTERS



INTERRUPT VECTOR REGISTER

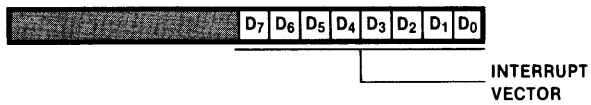


Figure 11. Special-Purpose Channel Registers

Table 4. Register Address Summary

Address (AD_7-AD_0)	(Hex)	Control Registers
X 0 1 1 1 0 0 X	38	Master Mode
X 0 1 0 1 1 1 X	2E	Command Channel 1
X 0 1 0 1 1 0 X	2C	Command Channel 2
		General-Purpose Channel Registers
X 0 0 1 1 0 1 X	1A	Current Address Register A-Channel 1, Segment/Tag
X 0 0 0 1 0 1 X	0A	Current Address Register A-Channel 1, Offset
X 0 0 1 1 0 0 X	18	Current Address Register A-Channel 2, Segment/Tag
X 0 0 0 1 0 0 X	08	Current Address Register A-Channel 2, Offset
X 0 0 1 0 0 1 X	12	Current Address Register 8-Channel 1, Segment/Tag
X 0 0 0 0 0 1 X	02	Current Address Register B-Channel 1, Offset
X 0 0 1 0 0 0 X	10	Current Address Register B-Channel 2, Segment/Tag
X 0 0 0 0 0 0 X	00	Current Address Register B-Channel 2, Offset
X 0 0 1 1 1 1 X	1E	Base Address Register A-Channel 1, Segment/Tag
X 0 0 0 1 1 1 X	0E	Base Address Register A-Channel 1, Offset
X 0 0 1 1 1 0 X	1C	Base Address Register A-Channel 2, Segment/Tag
X 0 0 0 1 1 0 X	0C	Base Address Register A-Channel 2, Offset
X 0 0 1 0 1 1 X	16	Base Address Register B-Channel 1, Segment/Tag
X 0 0 0 0 1 1 X	06	Base Address Register B-Channel 1, Offset
X 0 0 1 0 1 0 X	14	Base Address Register B-Channel 2, Segment/Tag
X 0 0 0 0 1 0 X	04	Base Address Register B-Channel 2, Offset
X 0 1 1 0 0 1 X	32	Current Operation Count Channel 1
X 0 1 1 0 0 0 X	30	Current Operation Count Channel 2
X 0 1 1 0 1 1 X	36	Base Operation Count Channel 1
X 0 1 1 0 1 0 X	34	Base Operation Count Channel 2
		Special-Purpose Channel Registers
X 1 0 0 1 0 1 X	4A	Pattern Channel 1
X 1 0 0 1 0 0 X	48	Pattern Channel 2
X 1 0 0 1 1 1 X	4E	Mask Channel 1
X 1 0 0 1 1 0 X	4C	Mask Channel 2
X 0 1 0 1 1 1 X	2E	Status Channel 1
X 0 1 0 1 1 0 X	2C	Status Channel 2
X 0 1 0 1 0 1 X	2A	Interrupt Save Channel 1
X 0 1 0 1 0 0 X	28	Interrupt Save Channel 2
X 1 0 1 1 0 1 X	5A	Interrupt Vector Channel 1
X 1 0 1 1 0 0 X	58	Interrupt Vector Channel 2
X 0 1 0 0 1 1 X	26	Chain Address, Channel 1 Segment/Tag
X 0 1 0 0 0 1 X	22	Chain Address, Channel 4 Offset
X 0 1 0 0 1 0 X	24	Chain Address, Channel 2 Segment/Tag
X 0 1 0 0 0 0 X	20	Chain Address, Channel 2 Offset
X 1 0 1 0 1 1 X	56	Channel Mode Channel 1 High
X 1 0 1 0 0 1 X	52	Channel Mode Channel 1 Low
X 1 0 1 0 1 0 X	54	Channel Mode Channel 2 High
X 1 0 1 0 0 0 X	50	Channel Mode Channel 2 Low

NOTE: X = ignored.

ADDRESSING

The address generated by the DTC is always a byte address, even though the memory is organized as 16-bit words. All word-sized data is word-aligned and must be addressed by even addresses (A_0 = 0). With byte transfers, the least significant address bit determines which half of the A/D bus is used for the transfer. An even address specifies the most significant byte (AD_8–AD_{15}), and an odd address specifies the least significant byte (AD_0–AD_7). This addressing mechanism applies to memory accesses as well as to I/O and Special I/O accesses.

COMMANDS

The Z8016 DTC responds to several commands that give the CPU direct control over operating parameters. The commands described below are executed immediately after being written by the CPU into the DTC's Command register. A summary of the DTC commands is given in Table 5.

Reset

The Reset command forces the DTC into an idle state, in which it waits for a Start Chain command. The Start Chain command initiates a chain operation on either channel.

Software Request

A channel's Software Request command initiates a previously programmed transfer. If both channels are active, Channel 1 has priority.

Set/Clear Hardware Mask

The Set/Clear Hardware Mask command sets or clears the Hardware Mask bit in the selected channel's Mode register.

Table 5. DTC Command Summary

Command	Opcode Bits 7654	Opcode Bits 3210	Example Code (HEX)
Reset	000X	XXXX	00
Start Chain Channel 1	101X	XXX0	A0
Start Chain Channel 2	101X	XXX1	A1
Clear Software Request Channel 1	010X	XX00	40
Clear Software Request Channel 2	010X	XX01	41
Set Software Request Channel 1	010X	XX10	42
Set Software Request Channel 2	010X	XX11	43
Clear Hardware Mask Channel 1	100X	XX00	80
Clear Hardware Mask Channel 2	100X	XX01	81
Set Hardware Mask Channel 1	100X	XX10	82
Set Hardware Mask Channel 2	100X	XX11	83
Clear CIE, IUS, IP Channel 1	001E	SP00	*
Clear CIE, IUS, IP Channel 2	001E	SP01	*
Set CIE, IUS, IP Channel 1	001E	SP10	*
Set CIE, IUS, IP Channel 2	001E	SP11	*
Clear Flip Bit Channel 1	011X	XX00	60
Clear Flip Bit Channel 2	011X	XX01	61
Set Flip Bit Channel 1	011X	XX10	62
Set Flip Bit Channel 2	011X	XX11	63

*NOTES:
1. E = Set to 1 to perform set/clear on CIE, Clear to 0 for no effect on CIE.
2. S = Set to 1 to perform set/clear on IUS, Clear to 0 for no effect on IUS.
3. P = Set to 1 to perform set/clear on IP, Clear to 0 for no effect on IP.
4. X = "don't care" bit. This bit is not decoded and may be 0 or 1.
5. Flip bit = reset to 0 for ARA = src, ARB = dst. Set to 1 for ARA = dst, ARB = src.

Set/Clear IP, IUS, and CIE

The Set/Clear IP, IUS, and CIE commands manipulate the Interrupt Control bits located in each channel's Status register. These bits implement the interrupt daisy-chain control. The IP, IUS, and CIE bits for each channel can be set and cleared individually or in combination.

Set/Clear Flip Bit

The Set/Clear Flip Bit command reverses the source and destination, thereby reversing the direction of data transfer without reprogramming the channel.

TIMING

The following descriptions and timing diagrams refer to the relative timing relationships of DTC signals during basic operations. For exact timing information, refer to the composite timing diagrams.

Bus Request And Acknowledge

Before the DTC can perform a DMA operation, it must gain control of the system bus. The $\overline{BUSREQ}$, $\overline{BAI}$, and $\overline{BAO}$ interface pins provide connections between the DTC and the host CPU and other DMA devices to arbitrate which device has control of the system bus. When the DTC wants to gain bus control, it drives $\overline{BUSREQ}$ Low. Bus Request and Acknowledge timing is shown in Figure 12.

Flowthrough Transactions

Timing for Flowthrough I/O and Flowthrough Memory transactions (Figures 13 and 14, respectively) is identical. There are two types of I/O space on the Z8016: I/O and Special I/O. Status lines ST_0–ST_3 specify when an I/O operation is being performed and which of the two I/O spaces is being accessed. During an I/O transaction, status signal N/$\overline{S}$ will be Low to indicate a System Level operation.

The timing for I/O operations is identical to the timing of Flowthrough memory transactions. An I/O cycle consists of three states: T_1, T_2, and T_3. The TWA state is a Wait state that can be inserted into the transaction cycle. The $\overline{AS}$ output is pulsed Low to mark the beginning of a T-cycle. The N/$\overline{S}$ line is set Low (System) and the R/$\overline{W}$ and B/$\overline{W}$ lines select Read or Write operations for bytes or words. The N/$\overline{S}$, R/$\overline{W}$ and B/$\overline{W}$ lines become stable during T_1 and remain stable until the end of T_3.

I/O address space is byte-addressed but both 8- and 16-bit data sizes are supported. During I/O transactions, the B/$\overline{W}$ output is High for byte transactions and Low for word transactions.

The R/$\overline{W}$ output is High during Read operations and Low during Write operations. $\overline{DS}$ is driven Low to signal the peripherals that data can be gated onto, or received from, the bus. $\overline{DS}$ is driven High to signal the end of the I/O transaction.

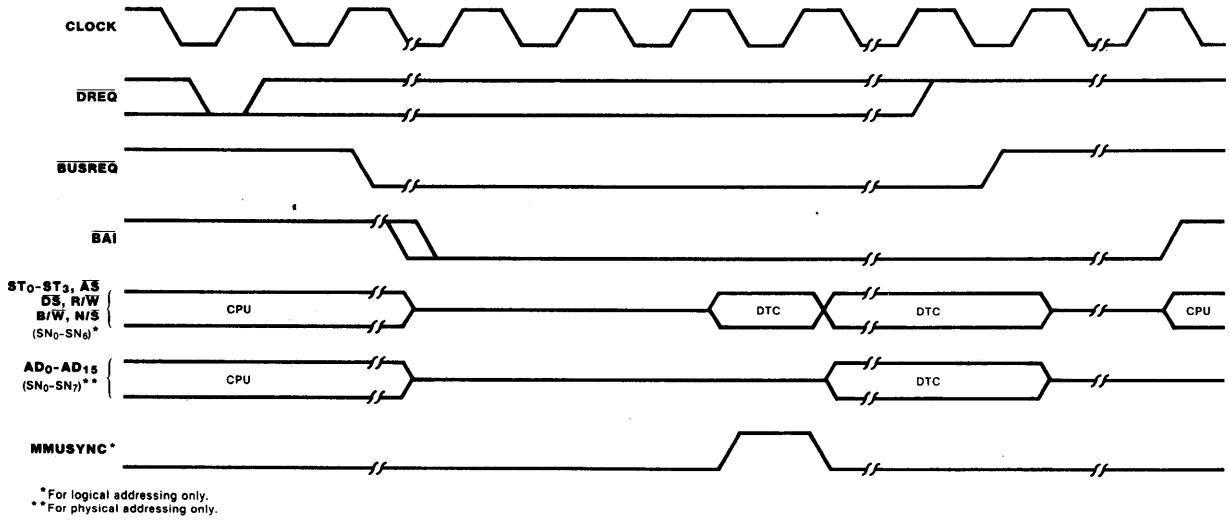
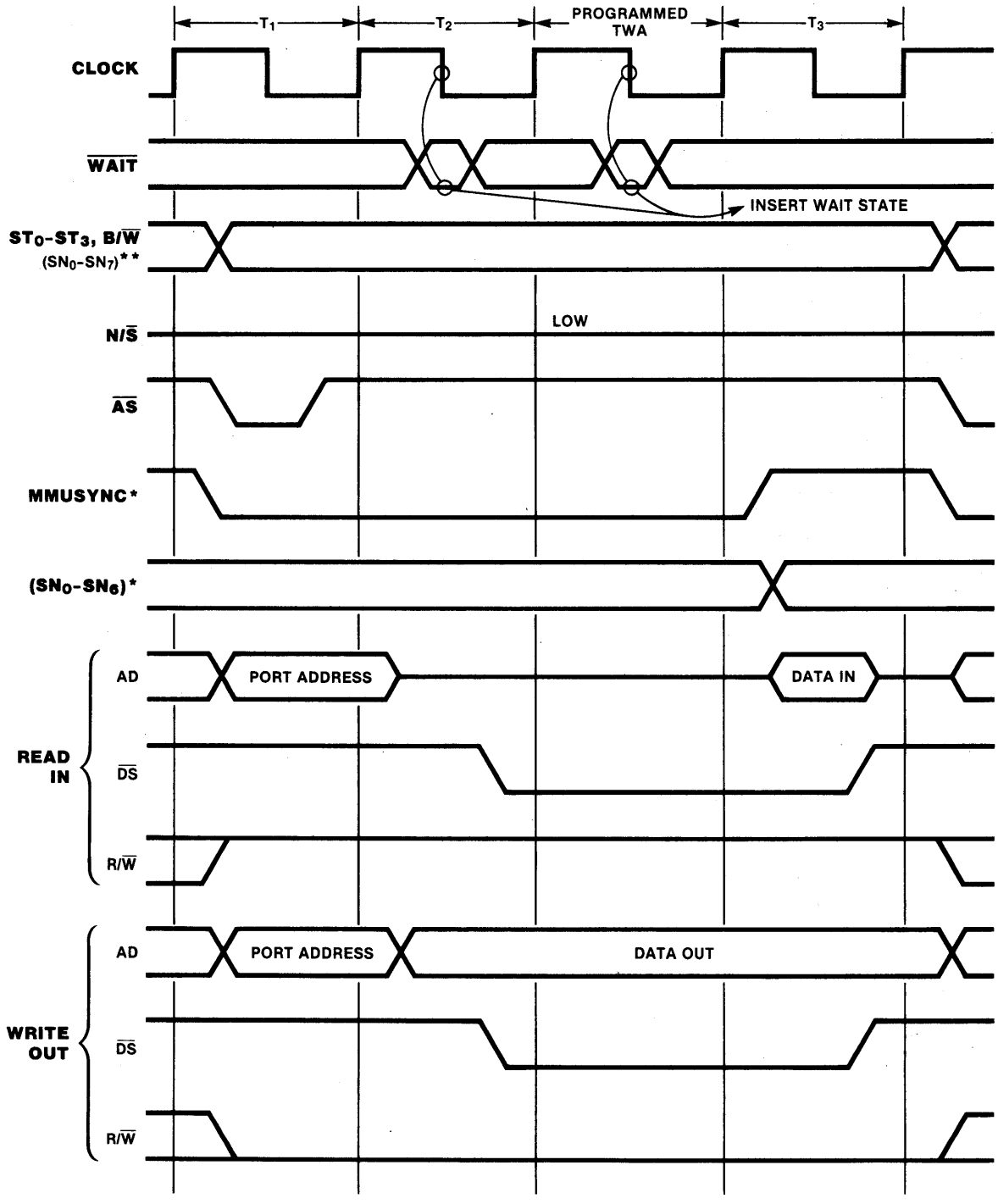
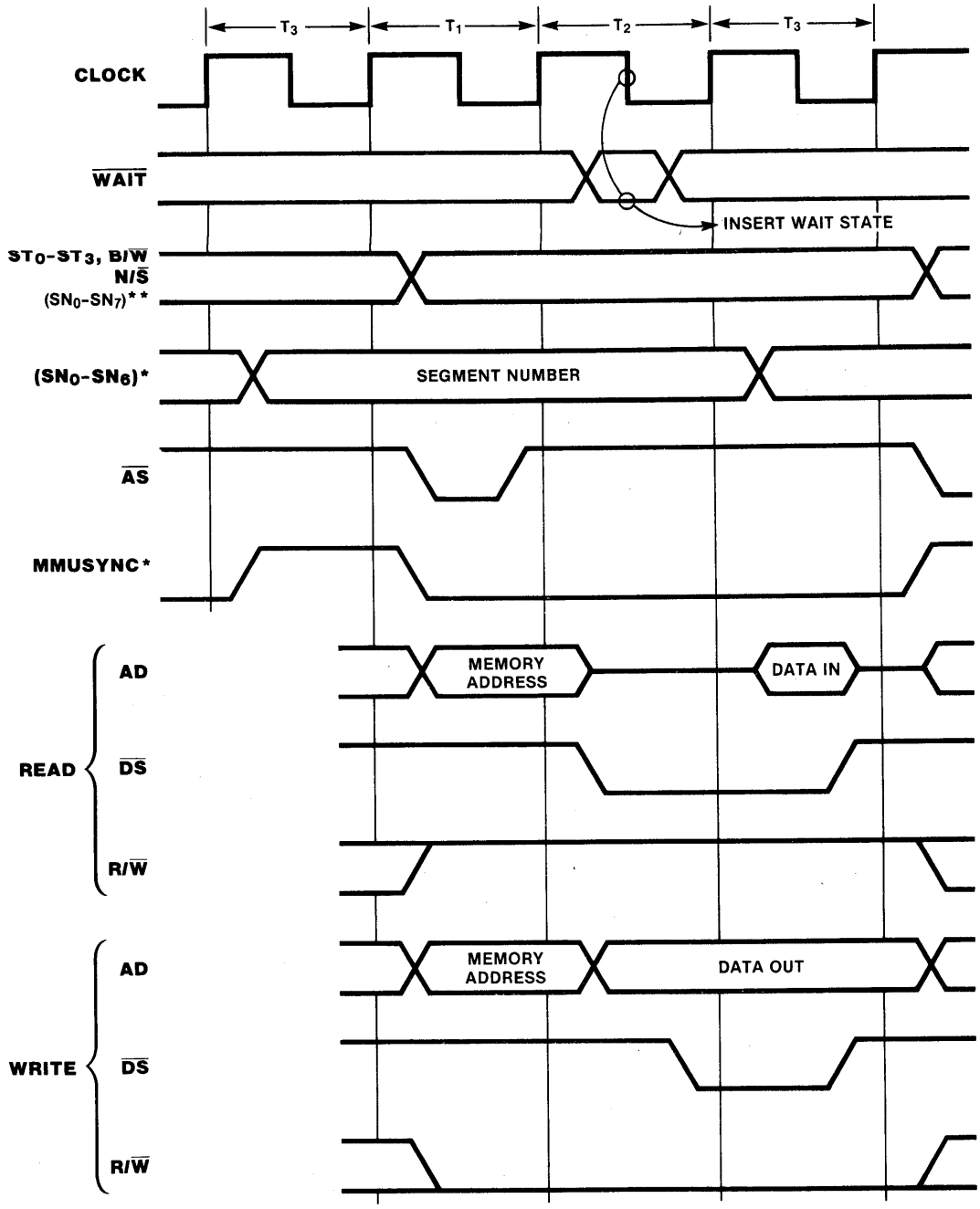


Figure 12. Bus Request and Acknowledge Timing



*For logical addressing only.
**For physical addressing only.

Figure 13. Flowthrough I/O Transaction Timing

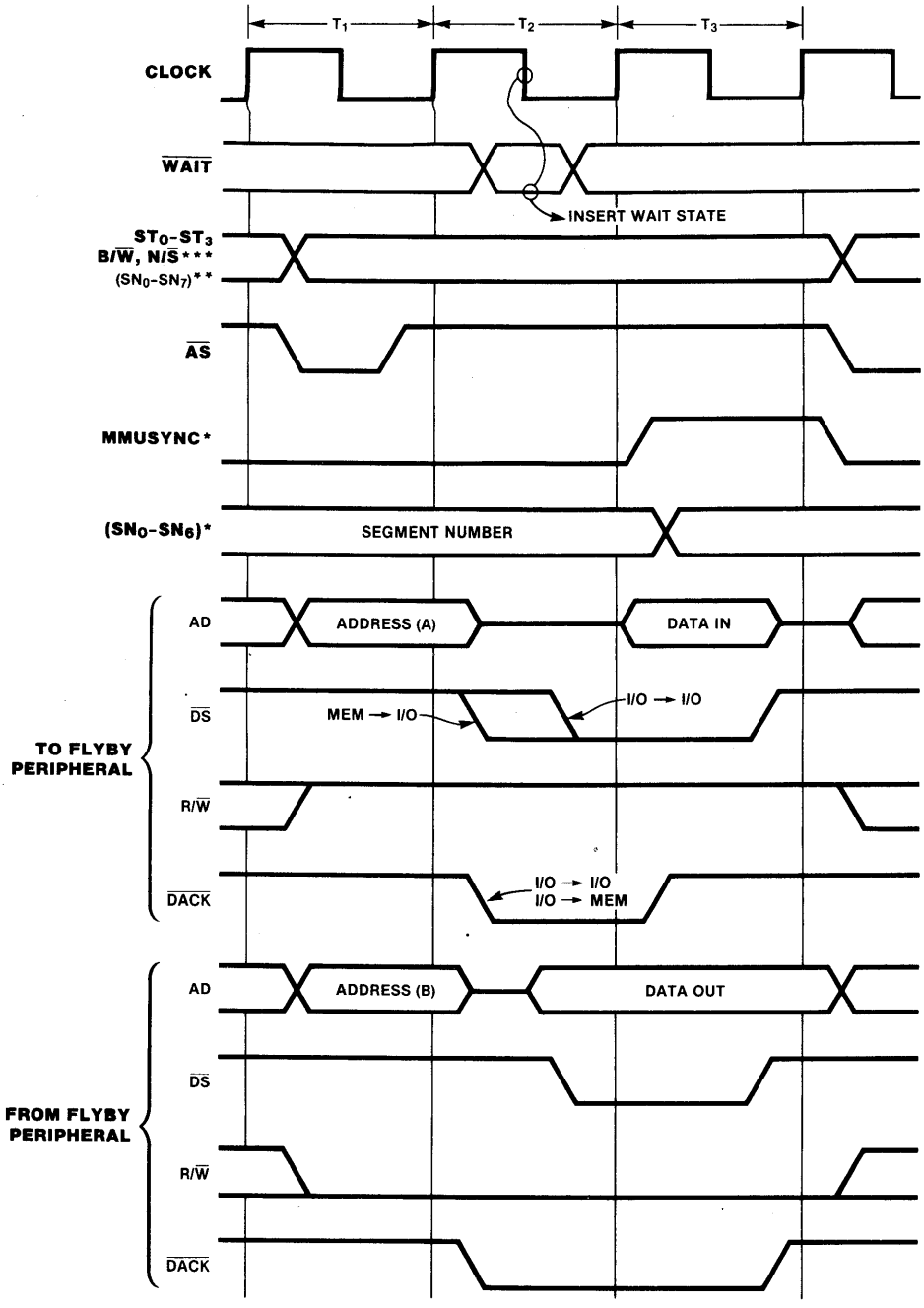


*For logical addressing only.
**For physical addressing only.

Figure 14. Flowthrough Memory Transaction Timing

Flyby Transactions

A Flyby operation is performed during three T-states. $\overline{AS}$ is pulsed during T_1 to signal the output of address information. R/$\overline{W}$ is High if the current ARA specifies source, and Low if the current ARB specifies destination. $\overline{DS}$ and $\overline{DACK}$ are driven active during T_2 to initiate the transfer, and driven inactive during T_3 to conclude the transfer. Wait states can be inserted between T_2 and T_3 to extend the active time to $\overline{DS}$ and $\overline{DACK}$. Flyby transaction timing is shown in Figure 15.



*Toggles for memory access in logical address space only.
**For physical addressing only.
***N/S will be low for I/O transactions.

(A) Address is current ARA
(B) Address is current ARB

Figure 15. Flyby Transaction Timing

$\overline{DREQ}$ Timing

The following section describes $\overline{DREQ}$ timing for various operations.

A High-to-Low transition of $\overline{DREQ}$ causes a single iteration of a DMA operation. A new transition can occur after the Low-to-High $\overline{AS}$ transition on the first memory or I/O access of the DMA iteration. Figure 16 shows the timing for a new transition to be applied and recognized to avoid giving up the bus at the end of the current iteration.

In Bus Hold mode, $\overline{DREQ}$ is sampled when a channel gains bus control. If $\overline{DREQ}$ is Low, an iteration of a DMA operation is performed. If $\overline{DREQ}$ is High, the channel retains bus control and continues to drive all bus control signals active or inactive, but performs no DMA operation.

In Demand mode during DMA operation, $\overline{DREQ}$ is sampled to determine whether the channel should perform another cycle or release the bus (Figure 17).

$\overline{DREQ}$ is sampled after each End of Chaining or Base-to-Current Reloading operation. If $\overline{DREQ}$ is active, the channel begins performing DMA operations immediately, without releasing the bus.

$\overline{DACK}$ Timing

During I/O and memory transactions, $\overline{WAIT}$ is sampled in the middle of T_2. If $\overline{WAIT}$ is High, and no programmable Wait states are selected, the DTC proceeds to T_3. Otherwise, one or more Wait states are inserted. $\overline{WAIT}$ is also sampled during T_{WA}. If $\overline{WAIT}$ is High the DTC proceeds to T_3, otherwise, additional Wait states are inserted. When both hardware and software Wait states are inserted, each $\overline{WAIT}$ time is sampled. A Low causes a hardware Wait state to be inserted in the next cycle. Software Wait state insertion is suspended until $\overline{WAIT}$ is High. Hardware Wait states can be inserted any time during the software Wait state sequence. $\overline{DACK}$ timing is shown in Figure 18.

$\overline{EOP}$ Timing

$\overline{EOP}$ is driven Low when a TC, MC, or $\overline{EOP}$ termination occurs. When a DMA operation has terminated, $\overline{EOP}$ is sampled on the falling edge of T_3 to determine if $\overline{EOP}$ has been driven Low. The generation of internal $\overline{EOP}$s and sampling of external $\overline{EOP}$s for Transfers-and-Searches follows the same timing used for Transfers. $\overline{EOP}$ timing is shown in Figure 19.

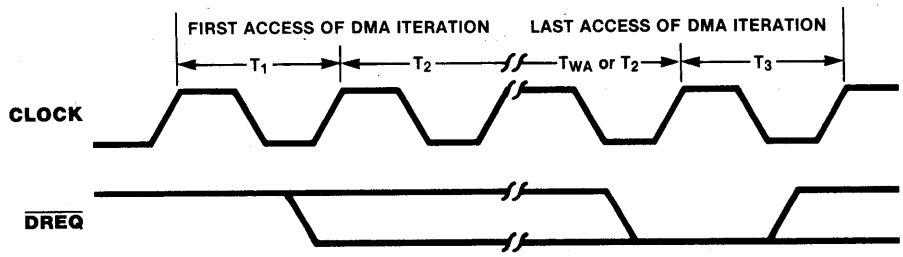
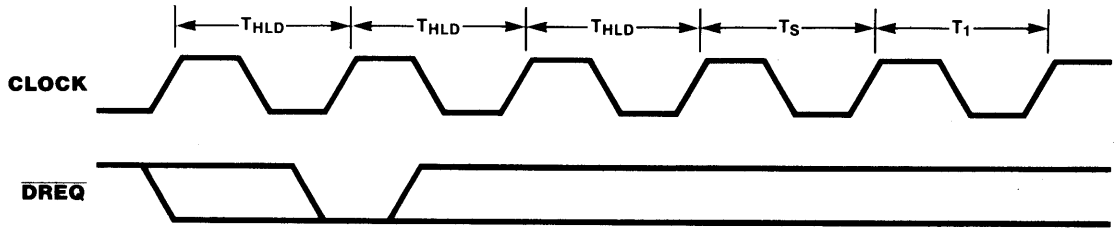
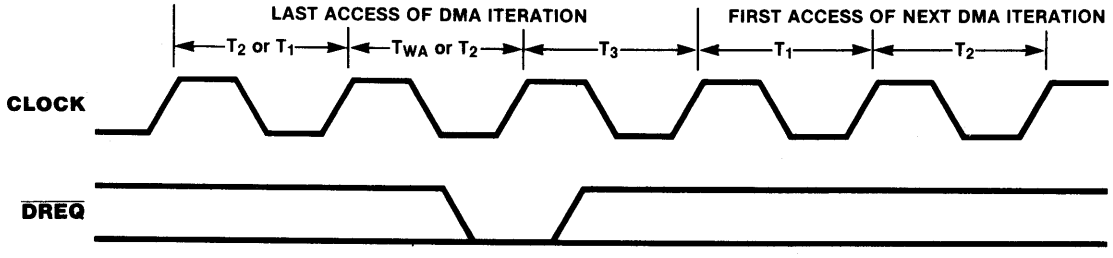


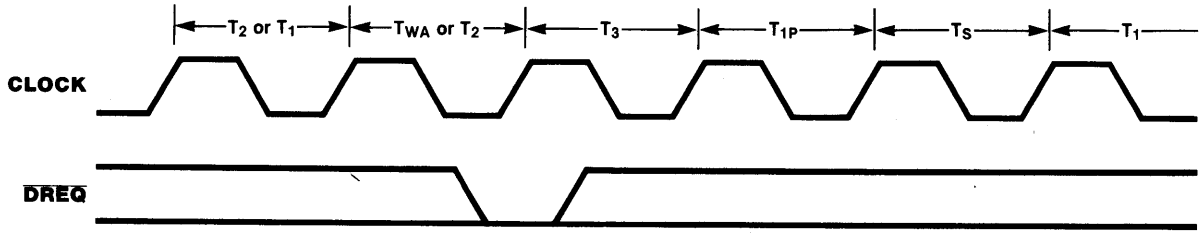
Figure 16. Sample $\overline{DREQ}$ During Single Transfer DMA Operations



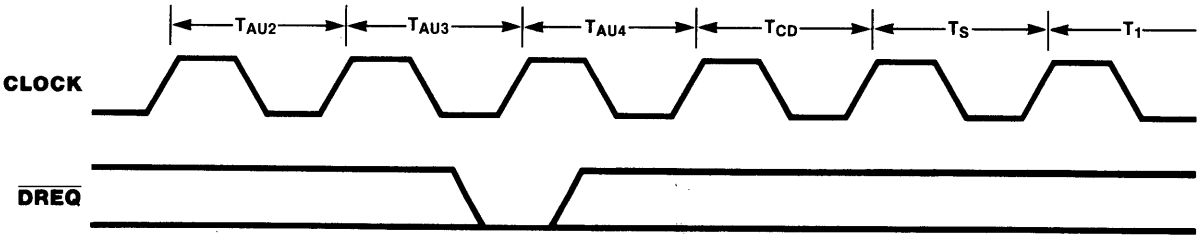
(A) Sampling of $\overline{DREQ}$ While in Bus Hold Mode



(B) $\overline{DREQ}$ Sampling in Demand Mode During DMA Operations



(C) Sampling $\overline{DREQ}$ at the End of Chaining



(D) Sampling $\overline{DREQ}$ at End of Base-to-Current Reloading

Figure 17. $\overline{DREQ}$ Sampling in Demand Mode

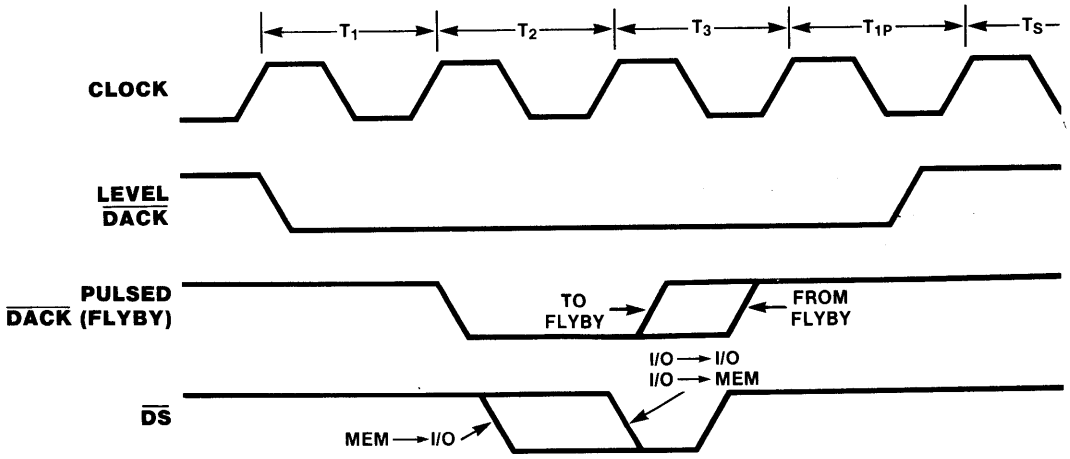


Figure 18. DACK Timing

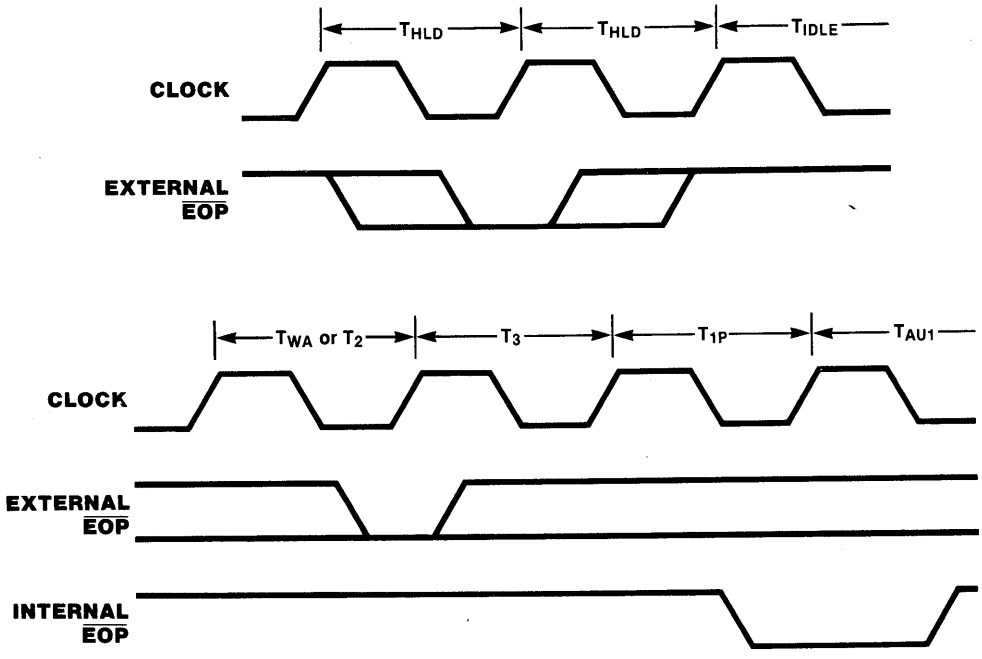
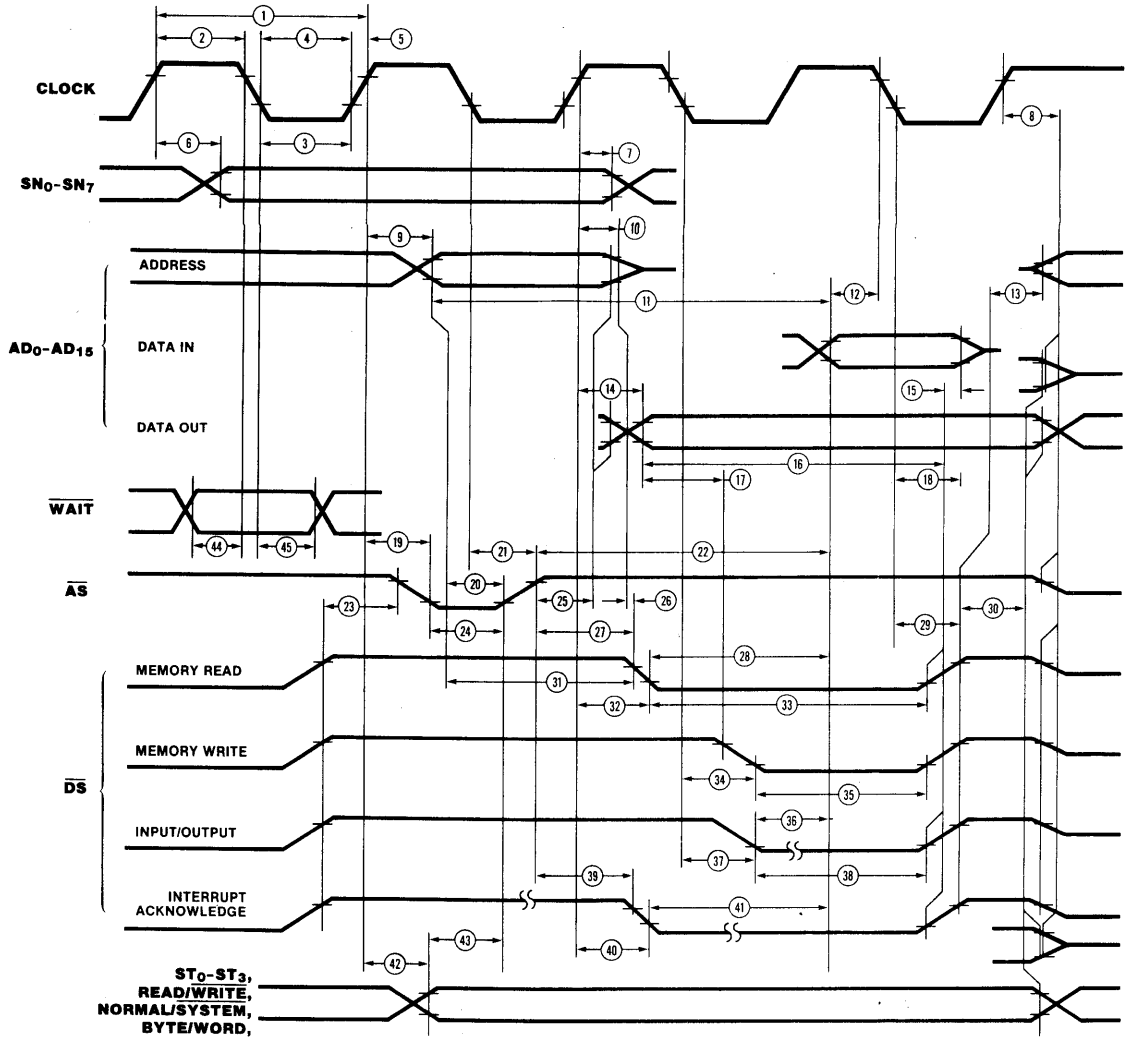


Figure 19. EOP Timing

ACTIVE STATE TIMING

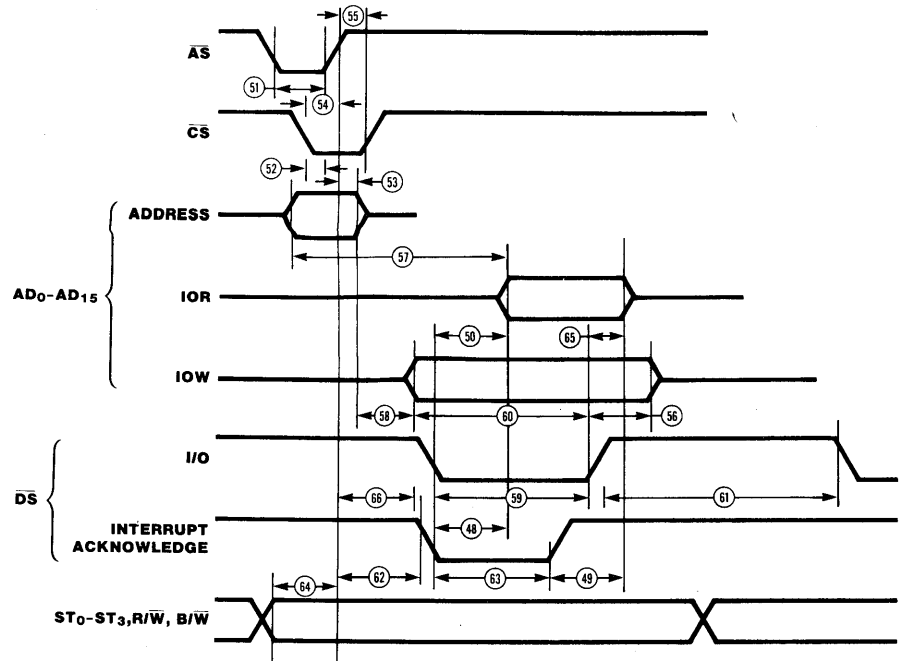


AC CHARACTERISTICS

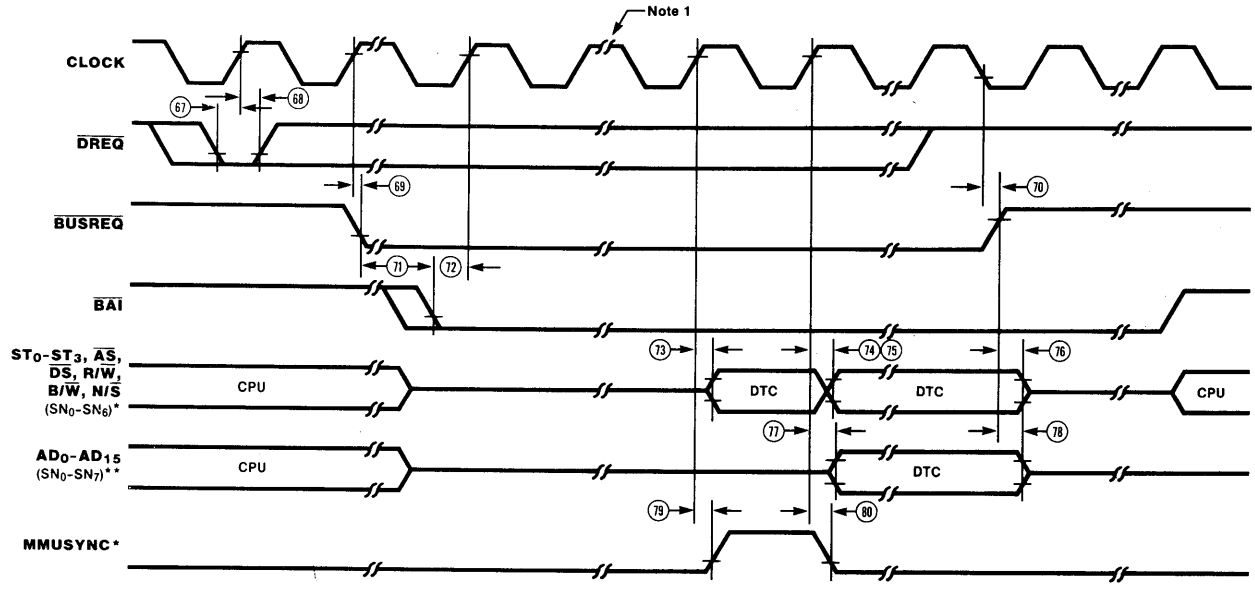
No.	Symbol	Description	Min (ns) (4 MHz)	Max (ns) (4 MHz)
1	TcC	Clock Cycle Time	250	2000
2	TwCh	Clock Width (High)	105	
3	TwCl	Clock Width (Low)	105	
4	TfC	Clock Fall Time		20
5	TrC	Clock Rise Time		20
6	TdC(SNv)	CLK ↑ to Segment Number Valid (50pF Load) Delay		110
7	TdC(SNn)	CLK ↑ to Segment Number Not Valid Delay	20	
8	TdC(Bz)	CLK ↑ to Bus Float Delay		65
9	TdC(A)	CLK ↑ to Address Valid Delay		90
10	TdC(Az)	CLK ↑ to Address Float Delay		65
11	TdA(DI)	Address Valid to Data in Required Valid Delay	400	
12	TsDI(C)	Data In to CLK ↓ Setup Time	20	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	80	
14	TdC(DO)	CLK ↑ to Data Out Valid Delay		90
15	ThDI(DS)	Data In to $\overline{DS}$ ↑ Hold Time	0	
16	TdDO(DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	230	
17	TdDO(SW)	Data out Valid to $\overline{DS}$ ↓ (Write) Delay	55	
19	TdC(ASf)	CLK ↑ to $\overline{AS}$ ↓ Delay		70
20	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50	
21	TdC(ASr)	CLK ↓ to $\overline{AS}$ ↑ Delay		70
22	TdAS(DI)	$\overline{AS}$ ↑ to Data In Required Valid Delay	300	
23	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	75	
24	TwAS	$\overline{AS}$ Width (Low)	80	
25	TdAS(A)	$\overline{AS}$ ↑ to Address Not Valid Delay	60	
26	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0	
27	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Read) Delay	75	
28	TdDSR(DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid Delay	165	
29	TdC(DSr)	CLK ↓ to $\overline{DS}$ ↑ Delay		60
30	TdDS(DO)	$\overline{DS}$ ↑ to Data Out (Write only) and Status Not Valid (Read and Write) Delay	85	
31	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	120	
32	TdC(DSR)	CLK ↑ to $\overline{DS}$ (Read) ↓ Delay		60
33	TwDSR	$\overline{DS}$ (Read) width (Low)	275	
34	TdC(DSW)	CLK ↓ to $\overline{DS}$ (Write) ↓ Delay		60
35	TwDSW	$\overline{DS}$ (Write) Width (Low)	160	
36	TdDSI(DI)	$\overline{DS}$ (Input) ↓ to Data in Required Valid Delay	325	
37	TdC(DSf)	CLK ↓ to $\overline{DS}$ (I/O) ↓ Delay		60
38	TwDS	$\overline{DS}$ (I/O) Width (Low)	160*	
39	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (ACK) Delay	100	
40	TdC(DS)	CLK ↑ to $\overline{DS}$ ↓ (ACK) Delay		100
41	TdSA(DI)	$\overline{DS}$ ↓ (ACK) to Data in Delay		150
42	TdC(S)	CLK ↑ to Status Valid Delay		110
43	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay		60
44	TsWT(C)	$\overline{WAIT}$ to CLK ↓ Setup Time	20	
45	ThWT(C)	$\overline{WAIT}$ to CLK ↓ Hold Time	10	

*Insert Wait states via software or hardware when accessing slow peripherals.

INACTIVE STATE TIMING



BUS EXCHANGE TIMING



*For logical addressing only.
**For physical addressing only.
Note 1: The DTC will begin driving the bus on the clock cycle following the clock cycle in which the set-up parameters are met.

AC CHARACTERISTICS (Continued)

No.	Symbol	Description	Min (ns) (4 MHz)	Max (ns) (4 MHz)
48	TdDSA(RDZ)	$\overline{DS}$ ↓ Acknowledge) to Read Data Float Delay		60
49	TdDSR(DOD)	$\overline{DS}$ ↑ (IOR)to Data Output Driven Delay		135
50	TdDSR(RDZ)	$\overline{DS}$ ↓ (IOR) to Read Data Float Delay		60
51	TwAS1	$\overline{AS}$ Low Width	70	
52	TsA(AS)	Address Valid to $\overline{AS}$ ↑ Setup Time	30	
53	ThAS(A)	$\overline{AS}$ ↑ to Address Not Valid Hold Time	40	
54	TsCS(AS)	$\overline{CS}$ Valid to $\overline{AS}$ ↑ Setup Time	0	
55	ThCS(AS)	$\overline{AS}$ ↑ to $\overline{CS}$ Not Valid Hold Time	40	
56	TdDS(Dn)	$\overline{DS}$ ↑ (IOW) to Data Not Valid Delay	25	
57	TdA(DRV)	Address Valid to Data (IOR) Required Valid Delay		540
58	TdA(DS)	Address Float to $\overline{DS}$ ↓ (IOR) Delay	0	
59	TwDS(IO)	$\overline{DS}$ (IO) Low Width	150	
60	TsD(DS)	Data Valid to $\overline{DS}$ ↑ Setup Time (IOW)	40	
61	TrDsh(DSR)	$\overline{DS}$ ↑ (IOW) to $\overline{DS}$ ↓ (IOW) (Write Recover Time applies only for issuing command)	4tcC	
62	TdASh(DSe)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (ACK) Delay	100	
63	TwDS(AK1)	$\overline{DS}$ (ACK) Width Low	150	
64	TsS(AS)	Status Valid to $\overline{AS}$ ↑ Setup Time	20	
65	TdDSW(Dn)	$\overline{DS}$ ↑ (IOR) to Data Not Valid Delay	25	
66	TdASh(DSI)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay (IO)	100	
67	TsDRQ(c)	$\overline{DREQ}$ Valid to CLK ↑ Setup Time	50	
68	ThDRQ(C)	CLK ↑ to $\overline{DREQ}$ Valid Hold Time	20	
69	TdC(BRQf)	CLK ↑ to $\overline{BUSREQ}$ ↓ Delay		150
70	TdC(BRQr)	CLK ↓ to $\overline{BUSREQ}$ ↑ Delay		150
71	TdBRQ(BAI)	$\overline{BUSREQ}$ ↓ to $\overline{BAI}$ ↓ Required Delay	0	
72	TsBAK(C)	$\overline{BAI}$ Valid to CLK ↑ Setup Time	50	
73	TdC(SNv)	CLK ↑ to Segment No. Valid (50pF Load) Delay		110
74	TdC(ASf)	CLK ↑ to $\overline{AS}$ ↓ Delay		70
75	TdC(S)	CLK ↑ to Status Valid Delay		110
76	TdBRQ(BUSc)	$\overline{BUSREQ}$ ↑ to Control Bus Float Delay		140
77	TdC(A)	CLK ↑ to Address Valid Delay		90
78	TdBRQ(BUSd)	$\overline{BUSREQ}$ ↑ to A/D Bus Float Delay		140
79	TdC(SNr)	CLK ↑ to SN7/MMUSYNC ↓ Delay**		110
80	TdC(SNf)	CLK ↑ to SN7/MMUSYNC ↓ Delay**	20	110

**Logical Addressing Only.

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs with respect to GND . . . . . −0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See ordering information

Storage Temperature . . . . . −65 °C to +150 °C

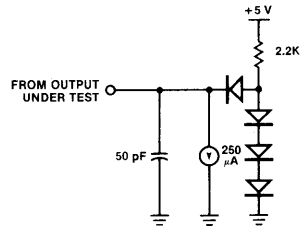
Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only: operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

STANDARD TEST CONDITIONS

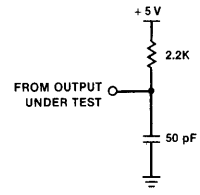
The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.



Standard Test Load



Open-Drain Test Load

DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	$V_{CC}-0.4$	$V_{CC}+0.3$	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10	µA	$0.4 \leq V_{IN} \leq V_{CC}$
I_{OL}	Output Leakage		±10	µA	$0.4 \leq V_{IN} \leq +V_{CC}$
I_{CC}	V_{CC} Supply Current		350	mA	$T_A = 0$ °C

NOTE: $V_{CC} = 5\ V \pm 5\%$ unless otherwise specified.

ORDERING INFORMATION

Product Number	Package/ Temp	Speed	Description
Z8016	CS	4.0 MHz	DTC (48-pin)
Z8016	PS	4.0 MHz	Same as above

Product Number	Package/ Temp	Speed	Description
Z8016A	CS	6.0 MHz	DTC (48-pin)
Z8016A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, P = Plastic; E = -40°C to +85°C, M = -55°C to +125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8030 Z8000™ Z-SCC Serial Communications Controller

Zilog

Product Specification

September 1983

Features

- Two independent, 0 to 1M bit/second, full-duplex channels, each with a separate crystal oscillator, baud rate generator, and Digital Phase-Locked Loop for clock recovery.
- Multi-protocol operation under program control; programmable for NRZ, NRZI, or FM data encoding.
- Asynchronous mode with five to eight bits and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Synchronous mode with internal or external character synchronization on one or two synchronous characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and SDLC Loop mode operation.
- Local Loopback and Auto Echo modes.

General Description

The Z8030 Z-SCC Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with the Zilog Z-Bus. The Z-SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The Z-SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, Digital Phase-Locked Loops, and crystal oscillators that dramatically reduce the need for external logic.

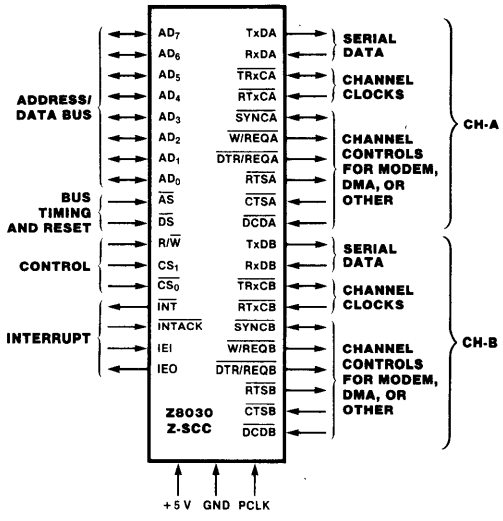


Figure 1. Pin Functions

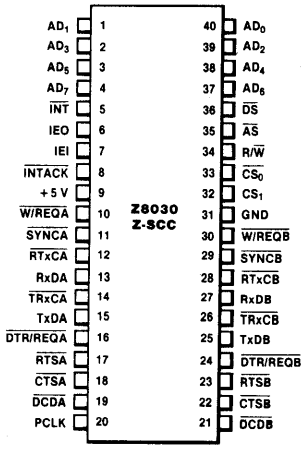


Figure 2. Pin Assignments

General Description (Continued)

The Z-SCC handles asynchronous formats, synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives, etc.).

The device can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The Z-SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-Bus daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

The Z8030 Z-SCC is packaged in a 40-pin ceramic DIP and uses a single +5 V power supply.

Pin Description

The following section describes the pin functions of the Z-SCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

AD_0-AD_7. *Address/Data Bus* (bidirectional, active High, 3-state). These multiplexed lines carry register addresses to the Z-SCC as well as data or control information to and from the Z-SCC.

$\overline{AS}$. *Address Strobe* (input, active Low). Addresses on AD_0–AD_7 are latched by the rising edge of this signal.

$\overline{CS_0}$. *Chip Select 0* (input, active Low). This signal is latched concurrently with the addresses on AD_0–AD_7 and must be active for the intended bus transaction to occur.

CS_1. *Chip Select 1* (input, active High). This second select signal must also be active before the intended bus transaction can occur. CS_1 must remain active throughout the transaction.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear to Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables their respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The Z-SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise-time signals. The Z-SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

$\overline{DS}$. *Data Strobe* (input, active Low). This signal provides timing for the transfer of data into and out of the Z-SCC. If $\overline{AS}$ and $\overline{DS}$ coincide, this is interpreted as a reset.

$\overline{DTR}$/$\overline{REQA}$, $\overline{DTR}$/$\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing a Z-SCC interrupt or the Z-SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the Z-SCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the Z-SCC interrupt daisy chain settles. When $\overline{DS}$ becomes active, the Z-SCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of $\overline{AS}$.

PCLK. *Clock* (input). This is the master Z-SCC clock used to synchronize internal signals. PCLK is not required to have any phase relationship with the master system clock, although the frequency of this clock must be at least 90% of the CPU clock frequency for a Z8000. PCLK is a TTL level signal.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RTxCA}$, $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock of the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 11) is set, the

Pin Description (Continued)

$\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

R/$\overline{W}$. *Read/Write* (input). This signal specifies whether the operation to be performed is a read or a write.

$\overline{SYNCA}$, $\overline{SYNCB}$. *Synchronization* (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit.

In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 10) but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{SYNC}$ must be driven Low two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{SYNC}$.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{TRxCA}$, $\overline{TRxCB}$. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{TRxC}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{W/REQA}$, $\overline{W/REQB}$. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the Z-SCC data rate. The reset state is Wait.

Functional Description

The functional capabilities of the Z-SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a Z8000 Family peripheral, it interacts with the Z8000 CPU and other peripheral circuits and is part of the Z-Bus interrupt structure.

Data Communications Capabilities. The Z-SCC provides two independent full-duplex channels programmable for use in any common Asynchronous or Synchronous data-communication protocol. Figure 3 and the following description briefly detail these protocols.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a

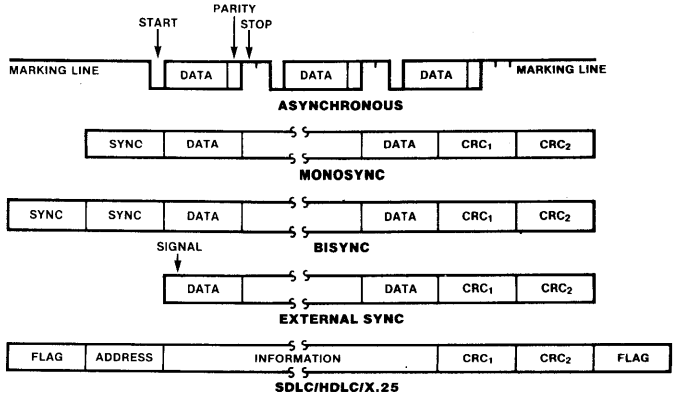


Figure 3. Some Z-SCC Protocols

Functional Description (Continued)

bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing or error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The Z-SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes. The Z-SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes, allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit synchronization pattern (Bisync), or with an external synchronization signal. Leading synchronous characters can be removed without interrupting the CPU.

Five- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the Z-SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 4.

CRC checking for Synchronous byte-oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all Synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The Z-SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The Z-SCC supports Synchronous bit-oriented protocols, such as SDLC and HDLC, by performing *automatic flag sending, zero insertion*, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the Z-SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. The Z-SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the Z-SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s.

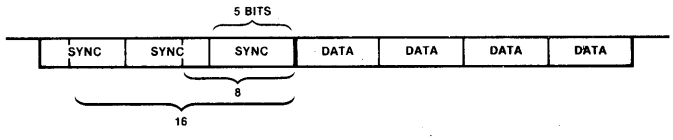


Figure 4. Detecting 5- or 7-Bit Synchronous Characters

Functional Description (Continued)

The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The Z-SCC can be conveniently used under DMA control to provide high-speed reception or transmission. In reception, for example, the Z-SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The Z-SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the Z-SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode. The Z-SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the Z-SCC performs the functions of a secondary station while a Z-SCC operating in regular SDLC mode can act as a controller (Figure 5).

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the Z-SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

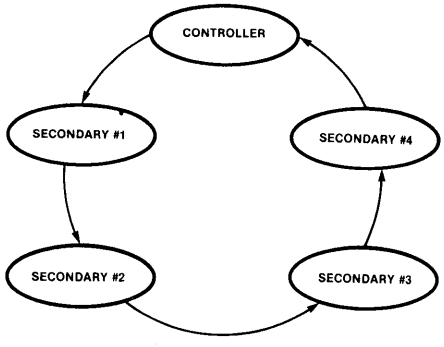


Figure 5. An SDLC Loop

Baud Rate Generator. Each channel in the Z-SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator may be echoed out via the $\overline{\text{TRxC}}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds):

$$\text{baud rate} = \frac{1}{2\,(\text{time constant} + 2) \times (\text{BR clock period})}$$

Digital Phase-Locked Loop. The Z-SCC contains a Digital Phase-Locked Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the Z-SCC receive clock, the transmit clock, or both.

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the

Functional Description (Continued)

incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{RTxC}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the Z-SCC via the $\overline{TRxC}$ pin (if this pin is not being used as an input).

Data Encoding The Z-SCC may be programmed to encode and decode the serial data in four different ways (Figure 6). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark) a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the Z-SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0 the bit is a 1.

Auto Echo and Local Loopback. The Z-SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{CTS}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{WAIT/REQUEST}$ on transmit.

The Z-SCC is also capable of Local Loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{CTS}$ and $\overline{DCD}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in Asynchronous, Synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The Z-SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the Z-SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be

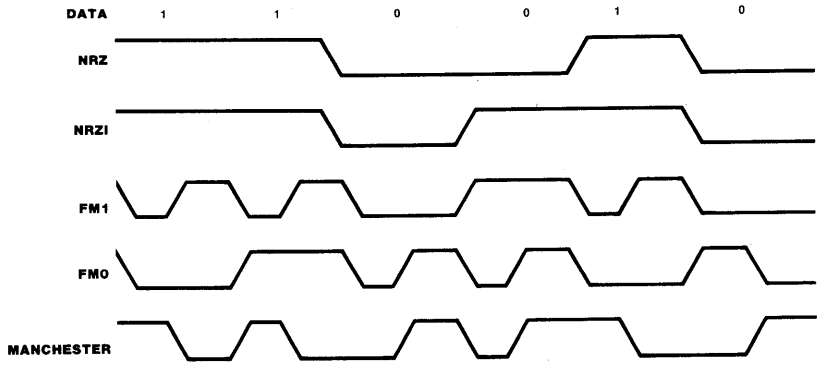


Figure 6. Data Encoding Methods

Functional Description (Continued)

read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. The Z-SCC interrupt scheme conforms to the Z-Bus specification. When a Z-SCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the A/D bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 10 and 11).

To speed interrupt response time, the Z-SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the Z-SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the Z-Bus interrupt priority chain (Figure 7). As a Z-Bus peripheral, the Z-SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the A/D bus.

In the Z-SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the Z-SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP is set two or three $\overline{AS}$ cycles after the interrupt condition occurs. Two or three $\overline{AS}$ rising edges are required from the time an interrupt condition occurs until $\overline{INT}$ is activated. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the Z-SCC and external to the Z-SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the Z-SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following: receiver overrun, framing error in Asynchronous mode, end-of-frame in SDLC mode and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$, and $\overline{SYNC}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count

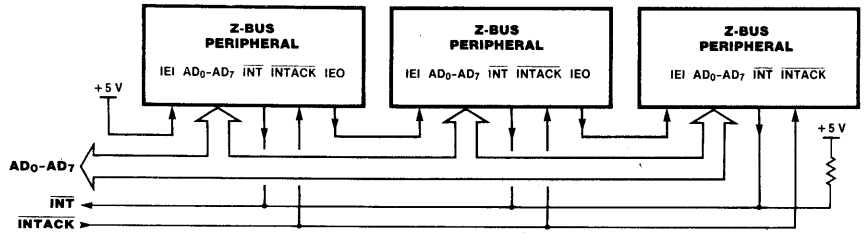


Figure 7. Z-BUS Interrupt Schedule

Functional Description (Continued)

in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the Z-SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer. The Z-SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT/REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT/REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the Z-SCC $\overline{\text{REQUEST}}$ output indicates that the Z-SCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the Z-SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR/REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The Z-SCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to the Zilog Z-Bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 8).

The logic for both channels provides

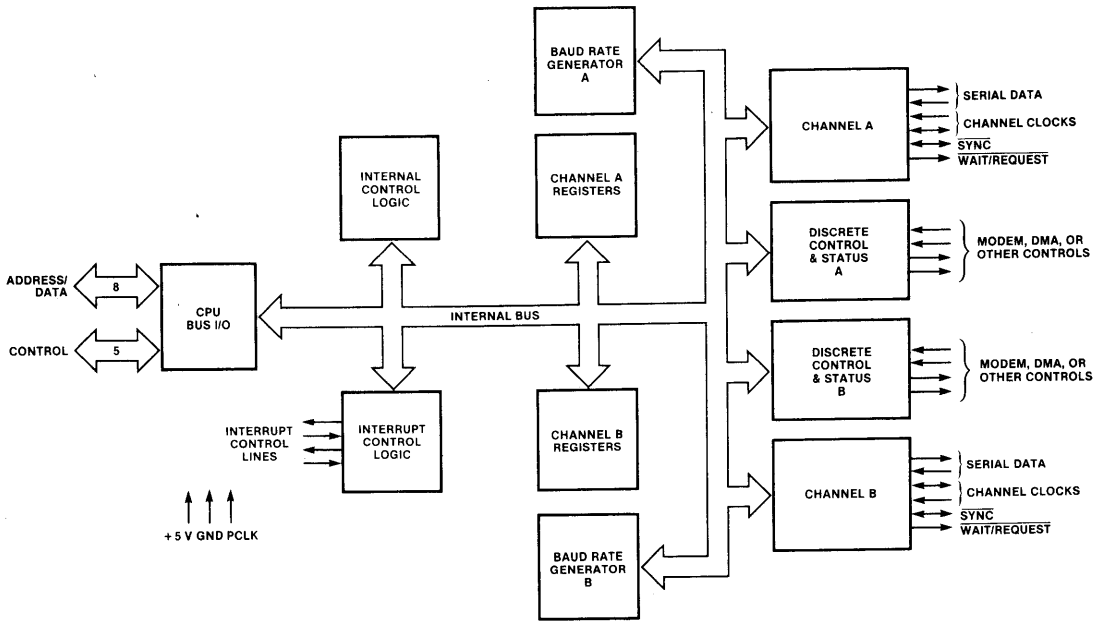


Figure 8. Block Diagram of Z-SCC Architecture

Architecture
(Continued)

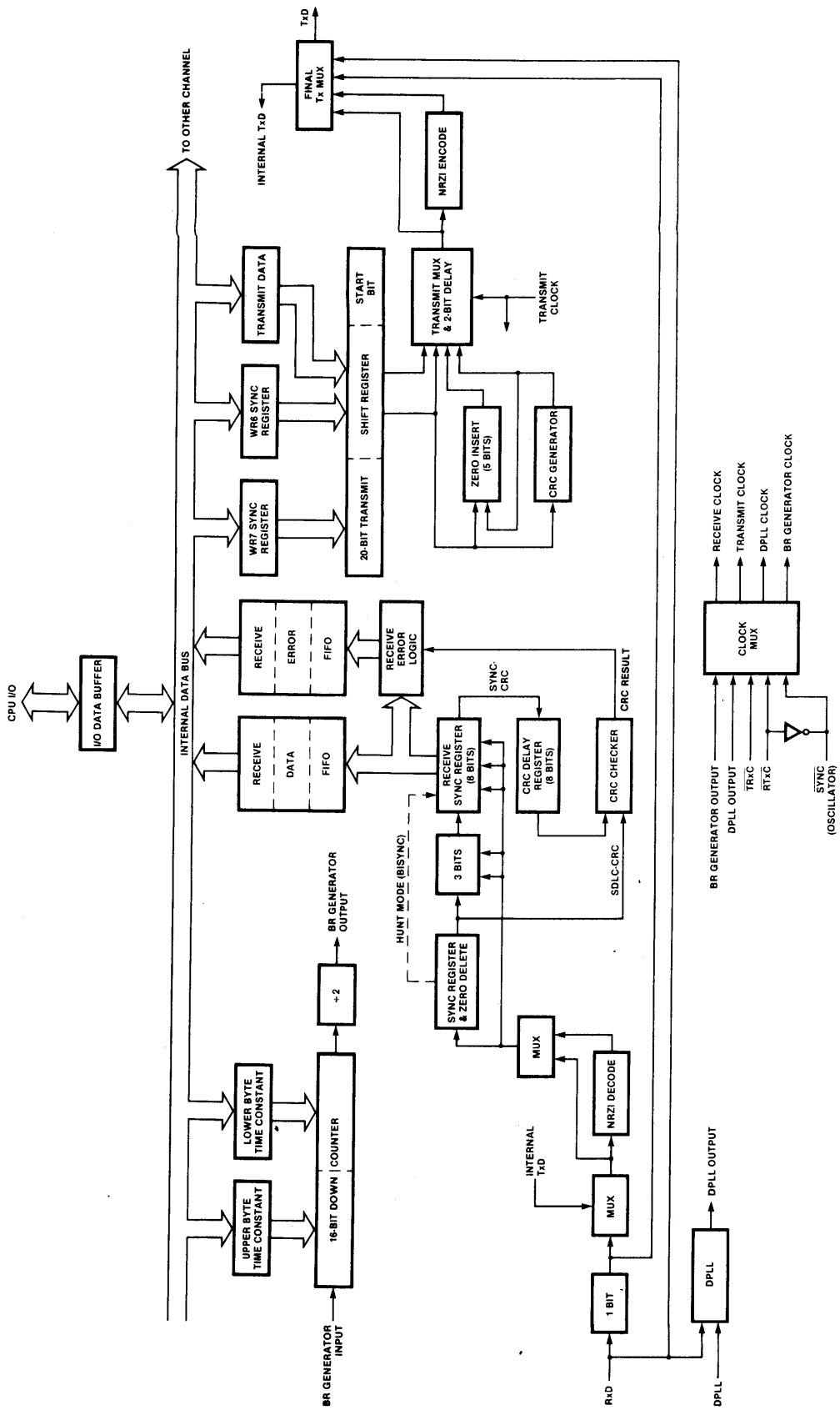


Figure 9. Data Path

2016-004

Architecture (Continued)

formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two sync character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write-only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0–WR15 — Write Registers 0 through 15.

RR0–RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The Z-SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 9 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and a 20-bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD)

Read Register Functions	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information

Write Register Functions	
WR0	CRC initialize, initialization commands for the various modes, shift right/shift left command
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync character or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The Z-SCC contains 13 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels. All of the registers in the Z-SCC are directly addressable. How the Z-SCC decodes the address placed on the address/data bus at the beginning of a Read or Write cycle is controlled by a command issued in WR0B. In the Shift Right mode the channel select A/$\overline{B}$ is taken from AD_0 and the state of AD_5 is ignored. In the Shift Left mode A/$\overline{B}$ is taken from AD_5 and the state of AD_0 is ignored. AD_7 and AD_6 are always ignored as address bits and the register address itself occupies AD_4–AD_1.

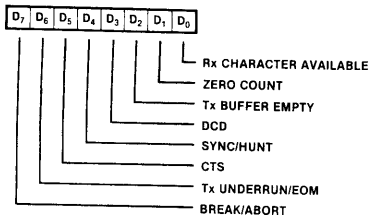
The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the Interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

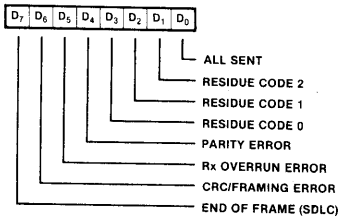
Read Registers. The Z-SCC contains eight read registers (actually nine, counting the receive buffer [RR8]) in each channel. Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 10 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

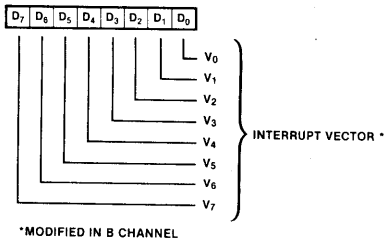
Read Register 0



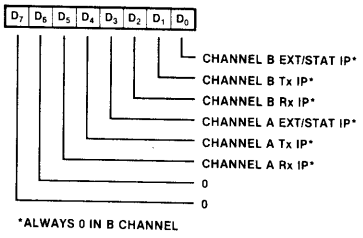
Read Register 1



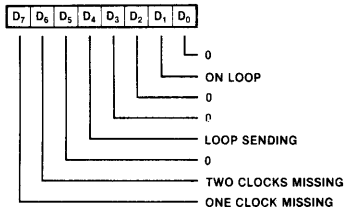
Read Register 2



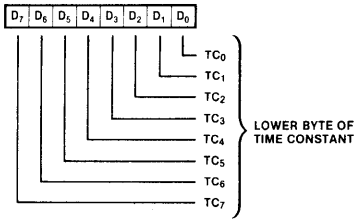
Read Register 3



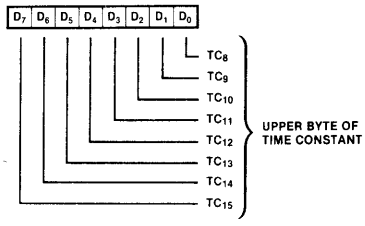
Read Register 10



Read Register 12



Read Register 13



Read Register 15

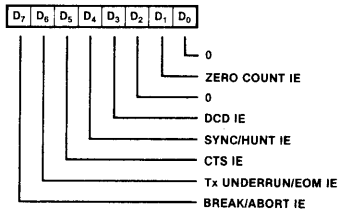
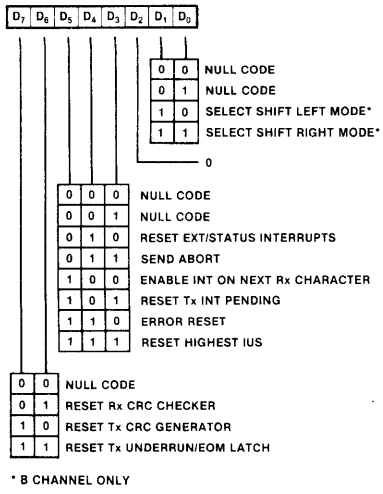


Figure 10. Read Register Bit Functions

Programming (Continued)

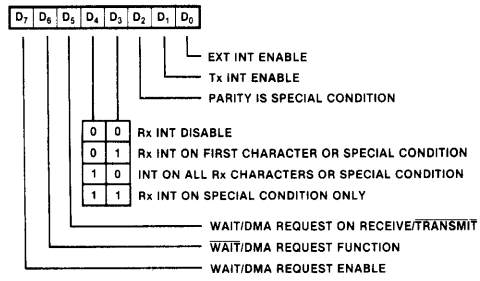
Write Registers. The Z-SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 11 shows the format of each write register.

Write Register 0

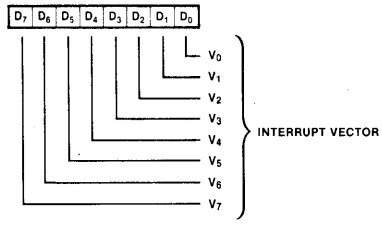


* B CHANNEL ONLY

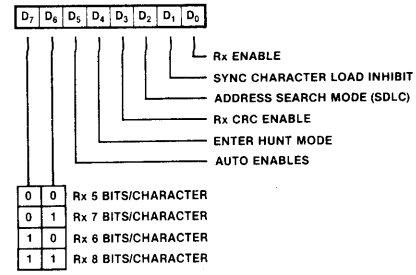
Write Register 1



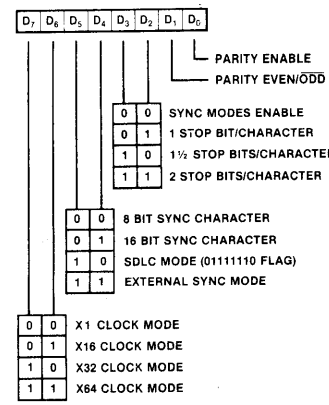
Write Register 2



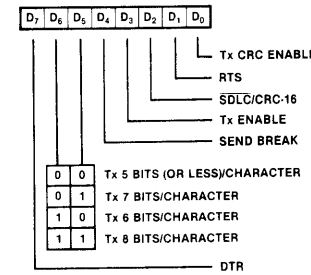
Write Register 3



Write Register 4



Write Register 5



Write Register 6

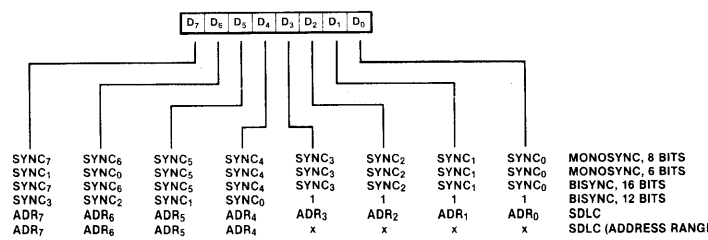
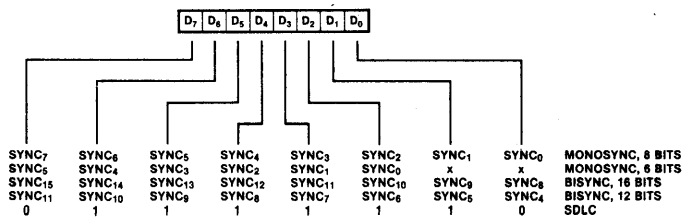
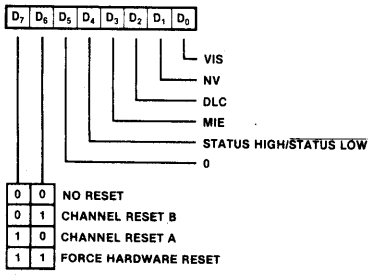


Figure 11. Write Register Bit Functions

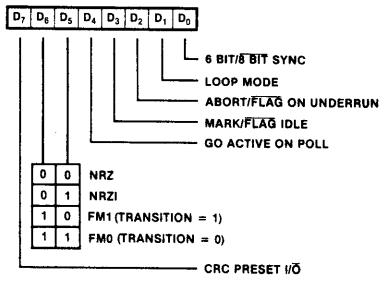
Write Register 7



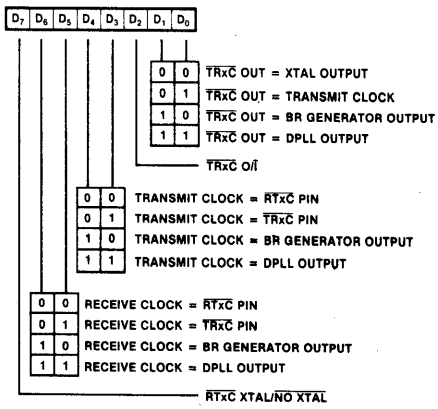
Write Register 9



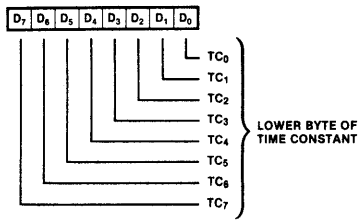
Write Register 10



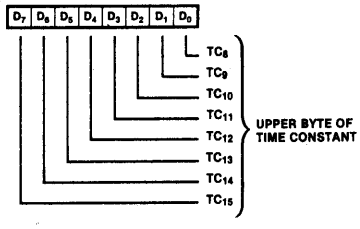
Write Register 11



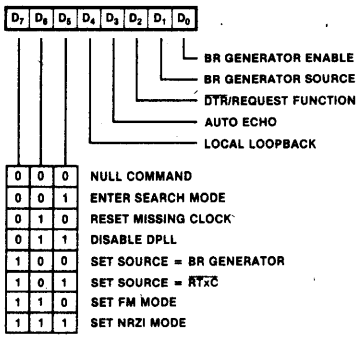
Write Register 12



Write Register 13



Write Register 14



Write Register 15

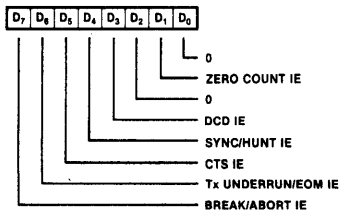


Figure 11. Write Register Bit Functions (Continued)

Timing

The Z-SCC generates internal control signals from $\overline{AS}$ and $\overline{DS}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{AS}$ and $\overline{DS}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the Z-SCC. The recovery time required for proper operation is specified from the rising edge of $\overline{DS}$ in the first transaction involving the Z-SCC to the falling edge of $\overline{DS}$ in the second transaction involving the Z-SCC. This time must be at least 6 PCLK cycles plus 200 ns.

Read Cycle Timing. Figure 12 illustrates read cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be High to indicate a Read cycle. CS_1 must also be High for the Read cycle to occur. The data bus drivers in the Z-SCC are then enabled while $\overline{DS}$ is Low.

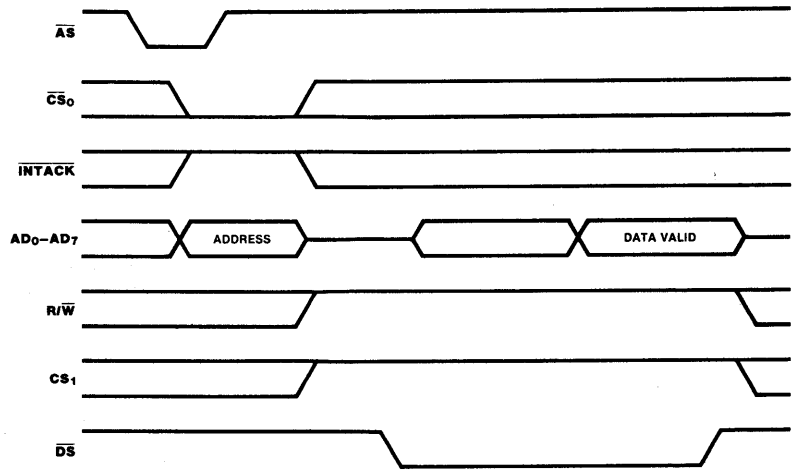


Figure 12. Read Cycle Timing

Write Cycle Timing. Figure 13 illustrates Write cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be Low to indicate a Write cycle. CS_1 must be High for the Write cycle to occur. $\overline{DS}$ Low strobes the data into the Z-SCC.

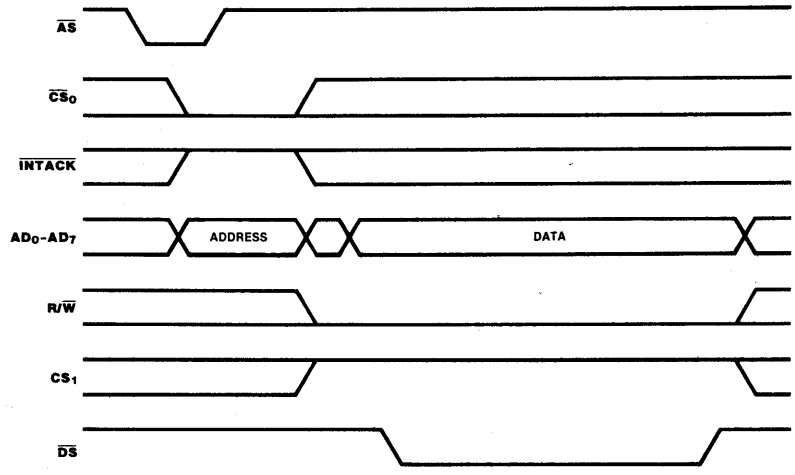


Figure 13. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 14 illustrates Interrupt Acknowledge cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. However, if $\overline{INTACK}$ is Low, the address and $\overline{CS_0}$ are ignored. The state of the R/$\overline{W}$ and CS_1 are also ignored for the duration of the Interrupt Acknowledge

Timing (Continued)

cycle. Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the Z-SCC and IEI is High when $\overline{DS}$ falls, the Acknowledge cycle was intended for the Z-SCC. In this case, the Z-SCC may be programmed to respond to $\overline{DS}$ Low by placing its interrupt vector on AD_0–AD_7. It then sets the appropriate Interrupt-Under-Service latch internally.

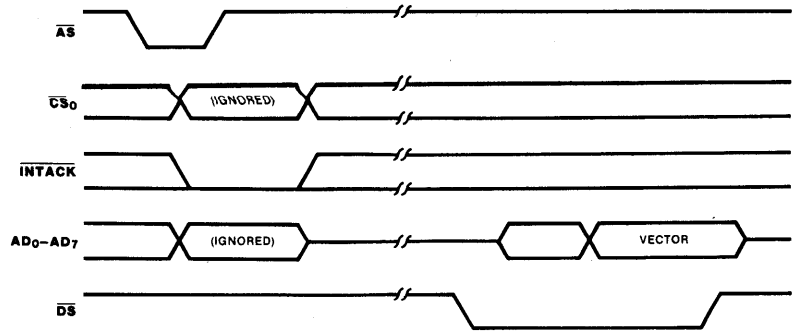


Figure 14. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \le V_{CC} \le +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

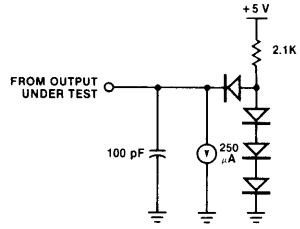


Figure 15. Standard Test Load

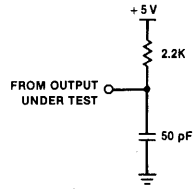


Figure 16. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \le V_{IN} \le +2.4V$
I_{OL}	Output Leakage		±10.0	µA	$0.4 \le V_{OUT} \le +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

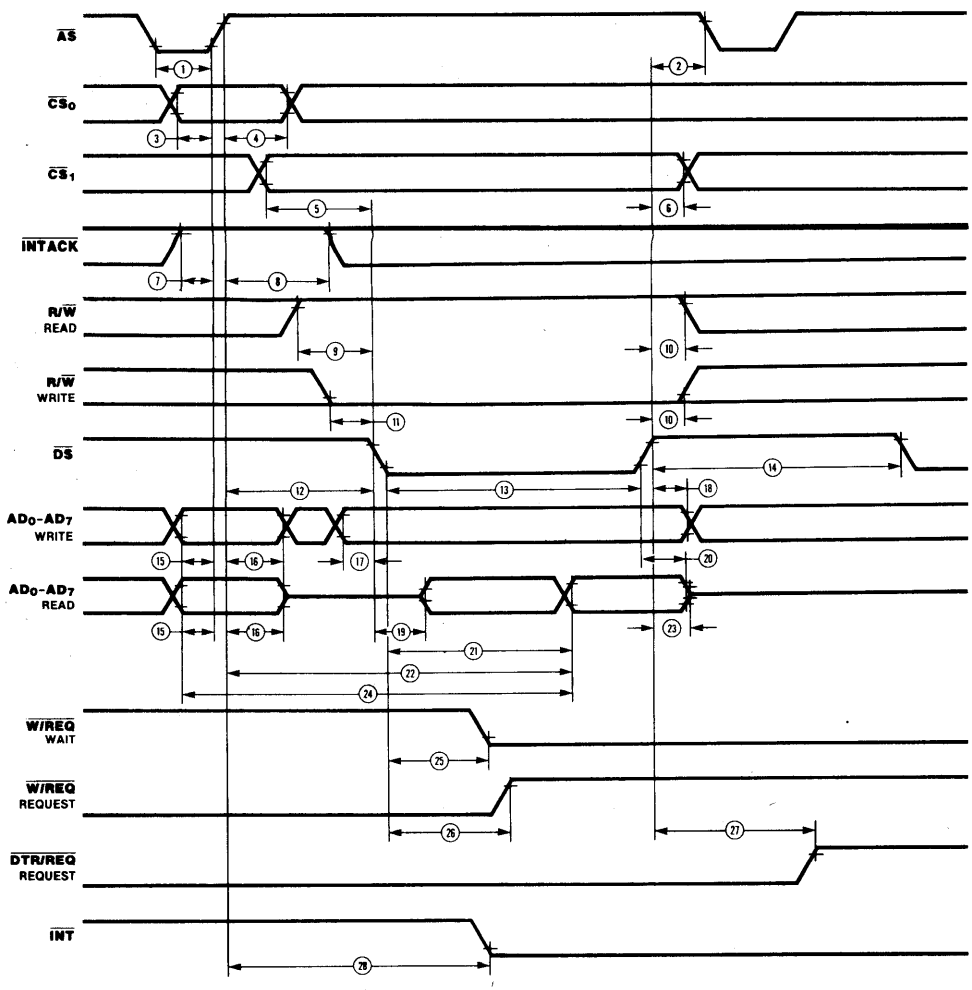
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

Read and Write Timing

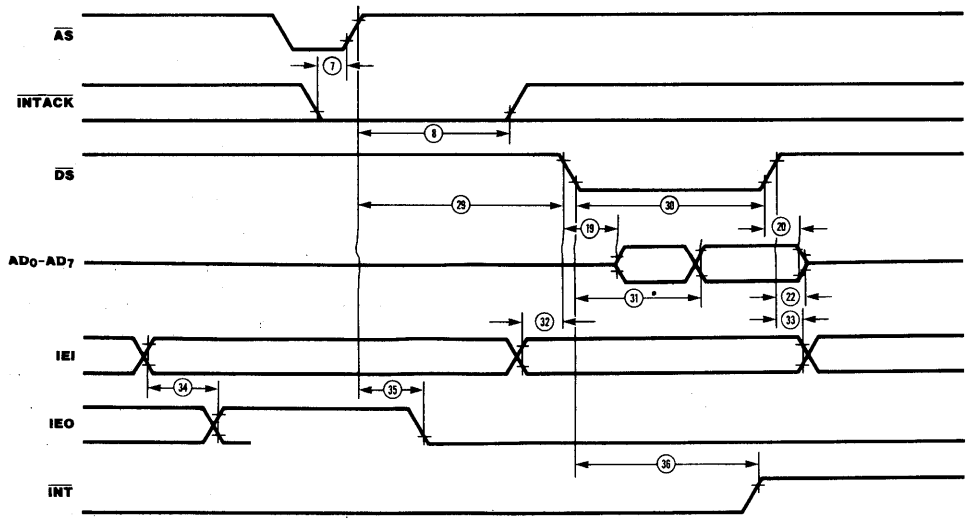


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwAS	$\overline{AS}$ Low Width	70		50		
2	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
3	TsCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
4	ThCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
5	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		80		1
6	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	55		40		1
7	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
8	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
10	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		
11	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
12	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		40		
13	TwDS1	$\overline{DS}$ Low Width	390		250		
14	TrC	Valid Access Recovery Time	6TcPC +200		6TcPC +130		2
15	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
16	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
17	TsDW(DS)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
18	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		
19	TdDS(DA)	$\overline{DS}$ ↓ to Data Active Delay	0		0		
20	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
21	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180	
22	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Valid Delay		520		335	

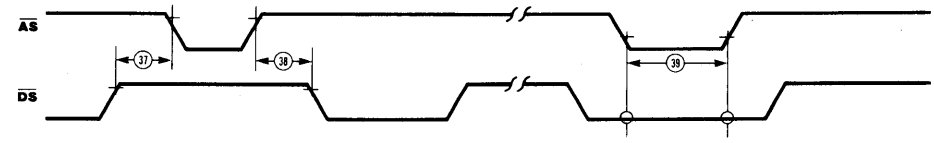
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Parameter applies only between transactions involving the SCC.

*Timings are preliminary and subject to change.
†Units in nanoseconds (ns).

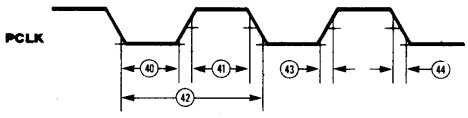
Interrupt Acknowledge Timing



Reset Timing



Cycle Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
23	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	3
24	TdA(DR)	Address Required Valid to Read Data Valid Delay		570		420	
25	TdDS(W)	$\overline{DS}$ ↓ to Wait Valid Delay		240		200	4
26	TdDSf(REQ)	$\overline{DS}$ ↓ to $\overline{W/REQ}$ Not Valid Delay		240		200	
27	TdDSr(REQ)	$\overline{DS}$ ↑ to $\overline{DTR/REQ}$ Not Valid Delay		5TcPC +300		5TcPC +250	
28	TdAS(INT)	$\overline{AS}$ ↑ to $\overline{INT}$ Valid Delay		500		500	4
29	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay	250		250		5
30	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	390		250		
31	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		250		180	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	120		100		
33	ThIEI(DSA)	IEI to $\overline{DS}$ ↑ (Acknowledge) Hold Time	0		0		
34	TdIEI(IEO)	IEI to IEO Delay		120		100	
35	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		250		250	6
36	TdDSA(INT)	$\overline{DS}$ ↓ (Acknowledge) to $\overline{INT}$ Inactive Delay		500		500	4
37	TdDS(ASQ)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay for No Reset	30		15		
38	TdASQ(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay for No Reset	30		30		
39	TwRES	$\overline{AS}$ and $\overline{DS}$ Coincident Low for Reset	250		250		7
40	TwPCl	PCLK Low Width	105	2000	70	1000	
41	TwPCh	PCLK High Width	105	2000	70	1000	
42	TcPC	PCLK Cycle Time	250	4000	165	2000	
43	TrPC	PCLK Rise Time		20		15	
44	TfPC	PCLK Fall Time		20		10	

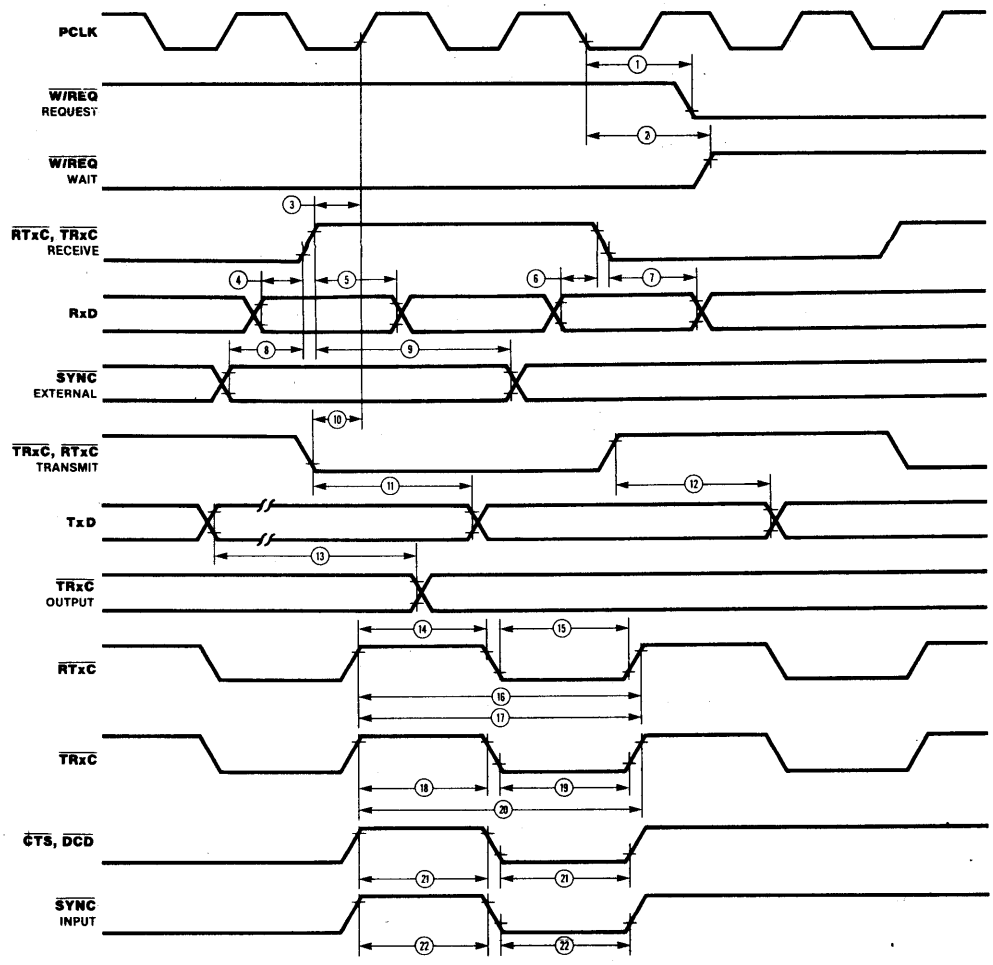
NOTES:

3. Float delay is defined as the time required for a ±0.5 V change in the output with a maximum dc load and minimum ac load.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any Z-SCC in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the highest priority device in the daisy chain, TsIEI(DSA) for the Z-SCC, and TdIEI(IEO) for each device separating them in the daisy chain.
6. Parameter applies only to a Z-SCC pulling $\overline{INT}$ Low at the beginning of the Interrupt Acknowledge transaction.
7. Internal circuitry allows for the reset provided by the Z8 to be recognized as a reset by the Z-SCC.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Units in nanoseconds (ns).

General Timing

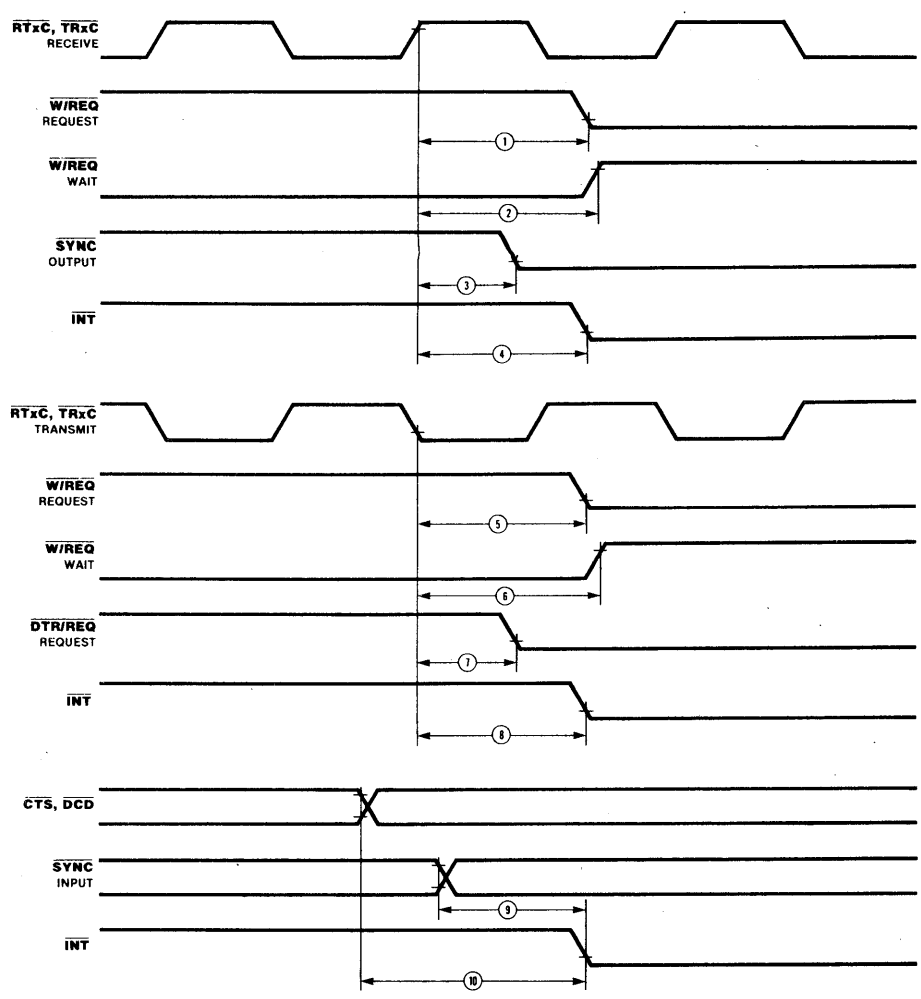


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdPC(REQ)	PCLK ↓ to W/REQ Valid		250		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350		350	
3	TsRXC(PC)	RxC ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	80	TwPCL	70	TwPCL	1,4
4	TsRXD(RXCr)	RxD to RxC ↑ Setup Time (X1 Mode)	0		0		1
5	ThRXD(RXCr)	RxD to RxC ↑ Hold Time (X1 Mode)	150		150		1
6	TsRXD(RXCf)	RxD to RxC ↓ Setup Time (X1 Mode)	0		0		1,5
7	ThRXD(RXCf)	RxD to RxC ↓ Hold Time (X1 Mode)	150		150		1,5
8	TsSY(RXC)	$\overline{SYNC}$ to RxC ↑ Setup Time	-200		-200		1
9	ThSY(RXC)	$\overline{SYNC}$ to RxC ↑ Hold Time	3TcPC +200		3TcPC +200		1
10	TsTXC(PC)	TxC ↓ to PCLK ↑ Setup Time	0		0		2,4
11	TdTXCf(TXD)	TxC ↓ to TxD Delay (X1 Mode)		300		300	2
12	TdTXCr(TXD)	TxC ↑ to TxD Delay (X1 Mode)		300		300	2,5
13	TdTXD(TRX)	TxD to TRxC Delay (Send Clock Echo)		200		200	
14	TwRTXh	RTxC High Width	180		180		6
15	TwRTXl	RTxC Low Width	180		180		6
16	TcRTX	RTxC Cycle Time	400		400		6
17	TcRTXX	Crystal Oscillator Period	250	1000	250	1000	3
18	TwTRXh	TRxC High Width	180		180		6
19	TwTRXl	TRxC Low Width	180		180		6
20	TcTRX	TRxC Cycle Time	400		400		6
21	TwEXT	DCD or CTS Pulse Width	200		200		
22	TwSY	$\overline{SYNC}$ Pulse Width	200				

NOTES:
1. RxC is RTxC or TRxC, whichever is supplying the receive clock.
2. TxC is TRxC or RTxC, whichever is supplying the transmit clock.
3. Both RTxC and SYNC have 30 pf capacitors to the ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between RxC and PCLK or TxC and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to chip PCLK requirements.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

System Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W}/\overline{REQ}$ Valid Delay	8	12	8	12	2,4
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	12	8	12	1,2,4
3	TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	4	7	4	7	2,4
4	TdRXC(INT)	$\overline{RxC}$ ↑ $\overline{INT}$ Valid Delay	8 +2	12 +3	8 +2	12 +3	1,2,4 5
5	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W}/\overline{REQ}$ Valid Delay	5	8	5	8	3,4
6	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	8	5	8	1,3,4
7	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DTR}/\overline{REQ}$ Valid Delay	4	7	4	7	3,4
8	TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	4 +2	6 +3	4 +2	6 +3	1,3,4 5
9	TdSY(INT)	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	2	3	2	3	1,5
10	TdEXT(INT)	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	3	2	3	1,5

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
4. Units equal to TcPC.
5. Units equal to $\overline{AS}$.

* Timings are preliminary and subject to change.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8030	CE	4.0 MHz	Z-SCC (40-pin)	Z8030A	CE	6.0 MHz	Z-SCC (40-pin)
Z8030	CM	4.0 MHz	Same as above	Z8030A	CM	6.0 MHz	Same as above
Z8030	CMB	4.0 MHz	Same as above	Z8030A	CMB	6.0 MHz	Same as above
Z8030	CS	4.0 MHz	Same as above	Z8030A	CS	6.0 MHz	Same as above
Z8030	DE	4.0 MHz	Same as above	Z8030A	DE	6.0 MHz	Same as above
Z8030	DS	4.0 MHz	Same as above	Z8030A	DS	6.0 MHz	Same as above
Z8030	PE	4.0 MHz	Same as above	Z8030A	PE	6.0 MHz	Same as above
Z8030	PS	4.0 MHz	Same as above	Z8030A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to 125°C, MB = -55°C to 125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8031 Z8000™ Z-ASCC Asynchronous Serial Communications Controller

Zilog

Product Specification

September 1983

Features

- Two independent, 0 to 1M bit/second, full-duplex channels, each with a separate crystal oscillator and baud rate generator.
- Programmable for NRZ, NRZI, or FM data encoding.
- Local Loopback and Auto Echo modes.
- Asynchronous communications with five to eight bits per character and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.

General Description

The Z8031 Z-ASCC Asynchronous Serial Communications Controller is a dual-channel data communications peripheral designed for use with the Zilog Z-BUS. The Z-ASCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators and crystal oscillators that dramatically reduce the need for external logic.

The Z-ASCC has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-BUS daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

The Z8031 Z-ASCC is packaged in a 40-pin ceramic DIP and uses a single +5 V power supply.

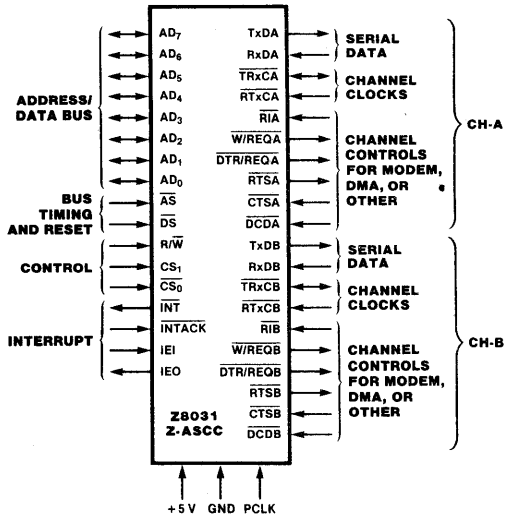


Figure 1. Pin Functions

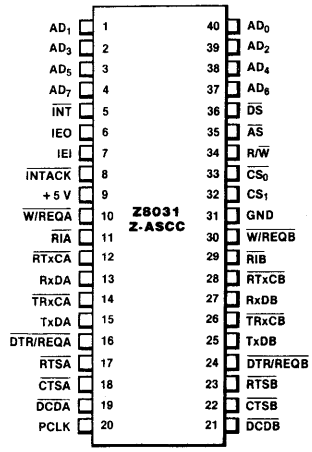


Figure 2. Pin Assignments

Pin Description

The following section describes the pin functions of the Z-ASCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

AD_0-AD_7. *Address/Data Bus* (bidirectional, active High, 3-state). These multiplexed lines carry register addresses to the Z-ASCC as well as data or control information to and from the Z-ASCC.

$\overline{AS}$. *Address Strobe* (input, active Low). Addresses on AD_0–AD_7 are latched by the rising edge of this signal.

$\overline{CS_0}$. *Chip Select 0* (input, active Low). This signal is latched concurrently with the addresses on AD_0–AD_7 and must be active for the intended bus transaction to occur.

CS_1. *Chip Select 1* (input, active High). This second select signal must also be active before the intended bus transaction can occur. CS_1 must remain active throughout the transaction.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear to Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables their respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The Z-ASCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise-time signals. The Z-ASCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

$\overline{DS}$. *Data Strobe* (input, active Low). This signal provides timing for the transfer of data into and out of the Z-ASCC. If $\overline{AS}$ and $\overline{DS}$ coincide, this is interpreted as a reset.

$\overline{DTR}/\overline{REQA}$, $\overline{DTR}/\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing a Z-ASCC interrupt or the Z-ASCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the Z-ASCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the Z-ASCC interrupt daisy chain settles. When $\overline{DS}$ becomes active, the Z-ASCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of $\overline{AS}$.

PCLK. *Clock* (input). This is the master Z-ASCC clock used to synchronize internal signals. PCLK is not required to have any phase relationship with the master system clock, although the frequency of this clock must be at least 90% of the CPU clock frequency for a Z8000. PCLK is a TTL level signal.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RIA}$, $\overline{RIB}$. *Ring Indicator* (inputs, active Low). These pins can act either as inputs or as part of the crystal oscillator circuit.

In normal operation (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Ring Indicator status bits in Read Register 0 (Figure 8) but have no other function.

$\overline{RTxCA}$, $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock of the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{RI}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 9) is set, the $\overline{RTS}$ signal goes Low. When the RTS bit is reset and Auto Enable is on, the signal goes High after the transmitter is empty. With Auto Enable off, the $\overline{RTS}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

R/$\overline{W}$. *Read/Write* (input). This signal specifies whether the operation to be performed is a read or a write.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

Pin Description (Continued)

TRxCA, TRxCB. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{TRxC}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

W/REQA, W/REQB. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the Z-ASCC data rate. The reset state is Wait.

Functional Description

The functional capabilities of the Z-ASCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a Z8000 peripheral, it interacts with the CPU and other peripheral circuits and is part of the system interrupt structure.

Data Communications Capabilities. The Z-ASCC provides two independent full-duplex channels programmable for use in any common Asynchronous data communication protocol. Figure 3 and the following description briefly detail this protocol.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The Z-ASCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

Baud Rate Generator. Each channel in the Z-ASCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{TRxC}$ pin, the output of the baud rate generator may be echoed out via the $\overline{TRxC}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds):

$$\text{time constant} = \frac{PCLK}{2\ (\text{clock factor})\ (\text{baud})} - 2$$

Digital Phase-Locked Loop. The Z-ASCC contains a Digital Phase-Locked Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the Z-ASCC receive clock, the transmit clock, or both.

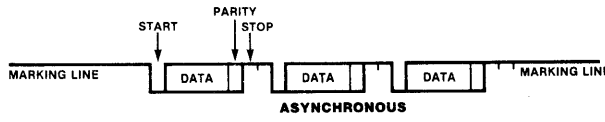


Figure 3. Z-ASCC Protocol

Functional Description (Continued)

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the Z-ASCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding The Z-ASCC may be programmed to encode and decode the serial data in four different ways (Figure 4). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark) a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the Z-ASCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0 the bit is a 1.

Auto Echo and Local Loopback. The Z-ASCC is capable of automatically echoing everything it receives. In Auto Echo mode, RxD is connected to TxD internally. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The Z-ASCC is also capable of Local Loopback. In this mode TxD is connected to RxD internally, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The Z-ASCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the Z-ASCC are automatically updated whenever any function is performed. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the

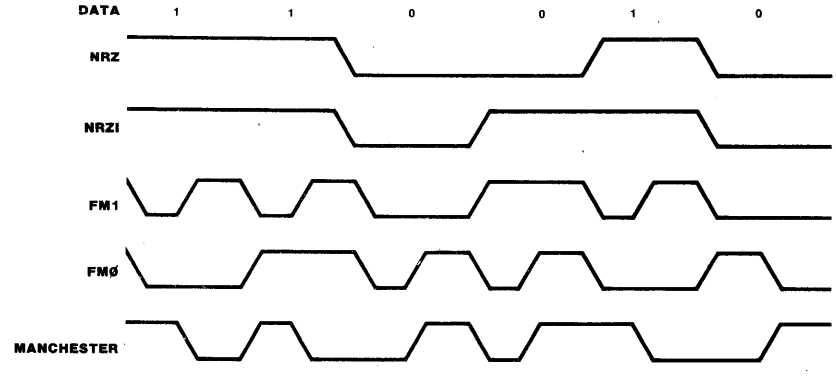


Figure 4. Data Encoding Methods

Functional Description (Continued)

Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. The Z-ASCC interrupt scheme conforms to the Z-BUS specification. When a Z-ASCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the A/D bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 8 and 9).

To speed interrupt response time, the Z-ASCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the Z-ASCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the Z-BUS interrupt priority chain (Figure 5). The Z-ASCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the A/D bus.

In the Z-ASCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the Z-ASCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP is set two or three $\overline{AS}$ cycles after the interrupt condition occurs. Two or three $\overline{AS}$ rising edges are required from the time an interrupt condition occurs until $\overline{INT}$ is activated. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the Z-ASCC and external to the Z-ASCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the Z-ASCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is receiver overrun, and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$, and $\overline{RI}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break.

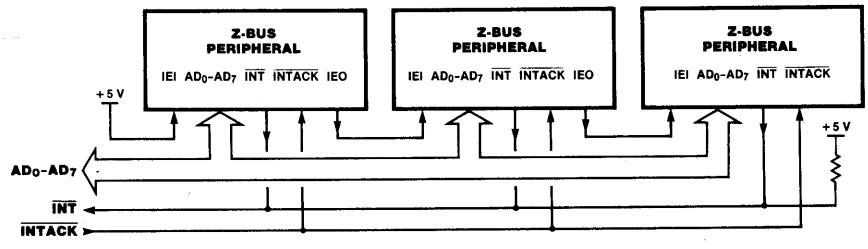


Figure 5. Z-BUS Interrupt Schedule

Functional Description (Continued)

CPU/DMA Block Transfer. The Z-ASCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}/\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the Z-ASCC $\overline{\text{REQUEST}}$ output indicates that the Z-ASCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the Z-ASCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}/\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The Z-ASCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to the Zilog Z-BUS. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 6).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the

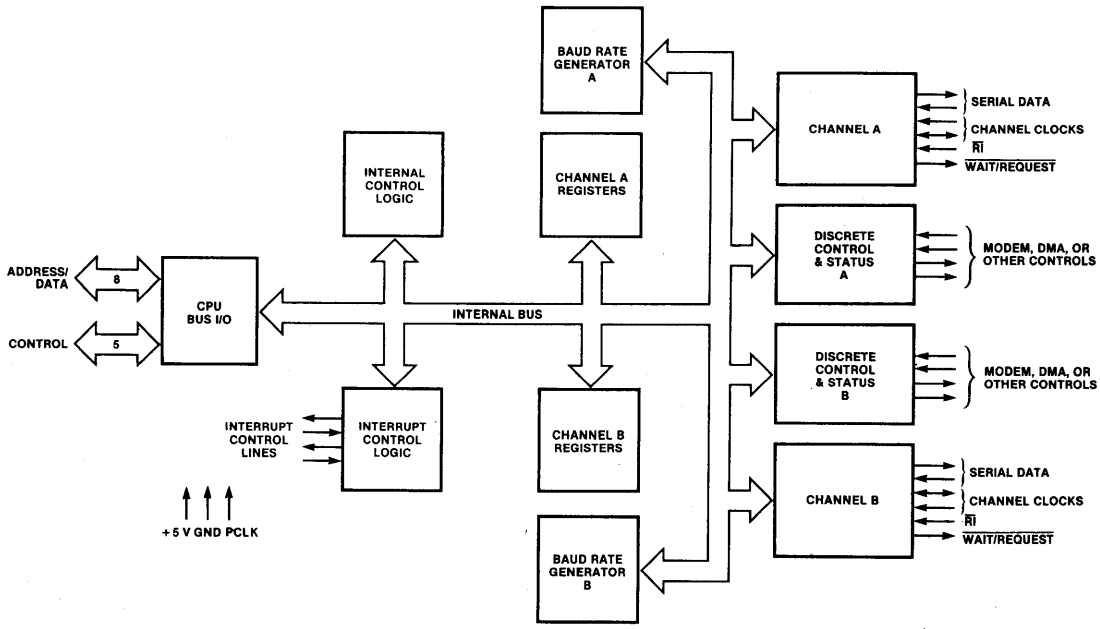


Figure 6. Block Diagram of Z-ASCC Architecture

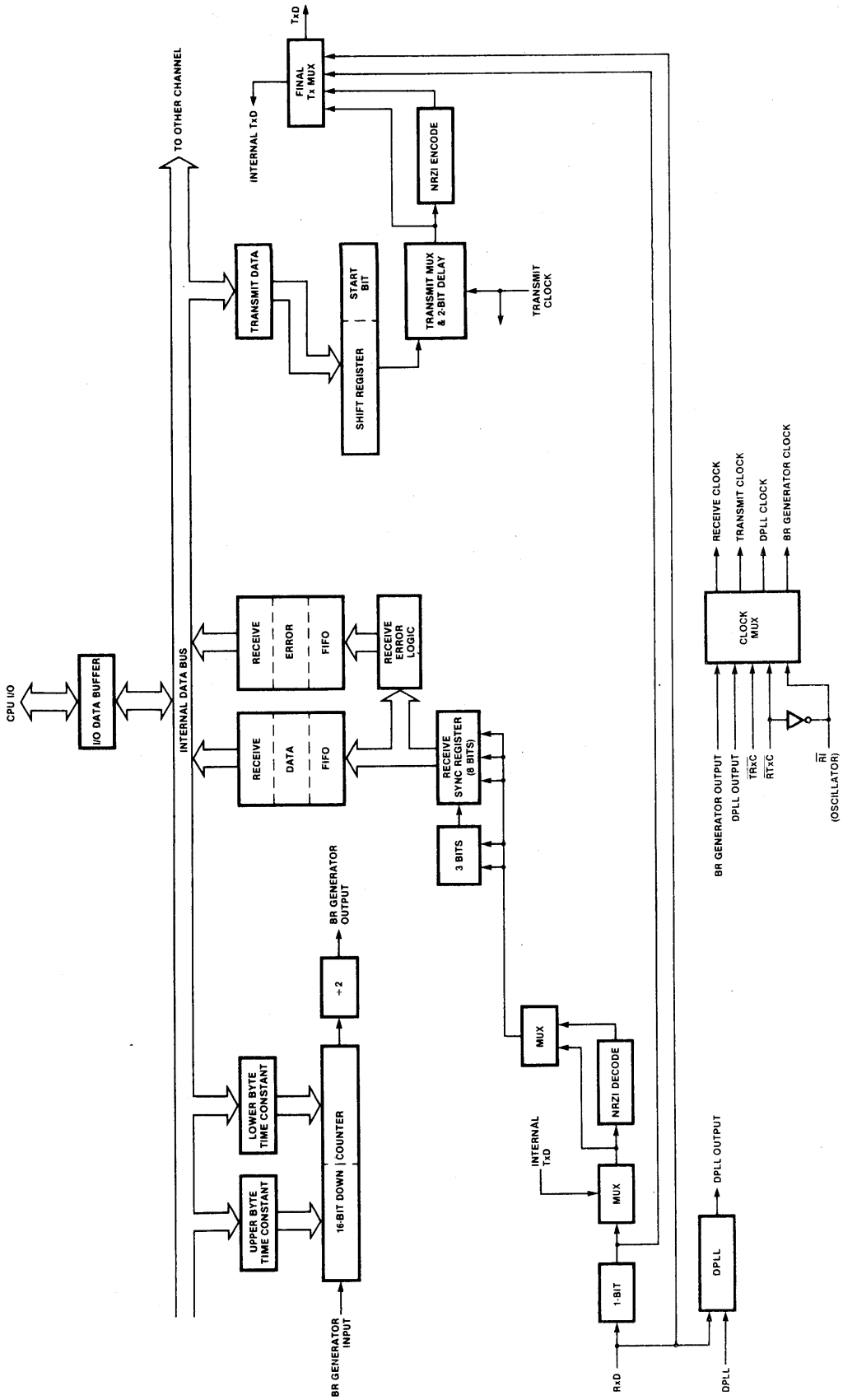


Figure 7. Data Path

Architecture (Continued)

baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write-only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0–WR15 — Write Registers 0-5, 8-15.

RR0–RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The Z-ASCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 7 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths depending on the selected mode (the character length determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and an 11-bit Transmit Shift register that is loaded from the Transmit Data register.

	Read Register Functions
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information
	Write Register Functions
WR0	CRC initialize, initialization commands for the various modes, shift right/shift left command
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The Z-ASCC contains 11 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels. All of the registers in the Z-ASCC are directly addressable. How the Z-ASCC decodes the address placed on the address/data bus at the beginning of a Read or Write cycle is controlled by a command issued in WR0B. In the shift right mode, the channel select A/$\overline{B}$ is taken from AD_0 and the state of AD_5 is ignored. In the shift left mode, A/$\overline{B}$ is taken from AD_5 and the state of AD_0 is ignored. AD_7 and AD_6 are always ignored as address bits and the register address itself occupies AD_4 - AD_1.

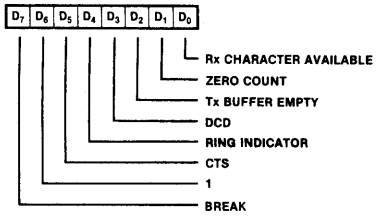
The system program first issues a series of commands to initialize the basic mode of operation. For example, the character length, clock rate, number of stop bits, even or odd parity might be set first. Then the Interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

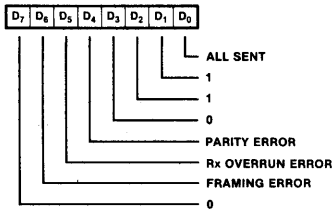
Read Registers. The Z-ASCC contains eight read registers (actually nine, counting the receive buffer [RR8]) in each channel. Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 8 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

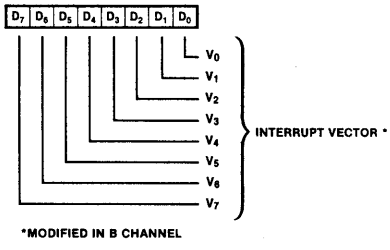
Read Register 0



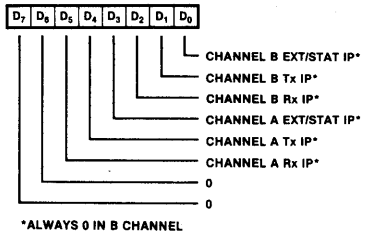
Read Register 1



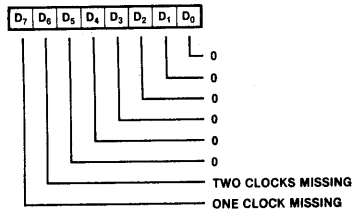
Read Register 2



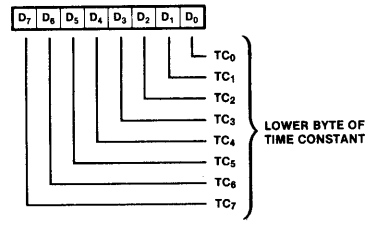
Read Register 3



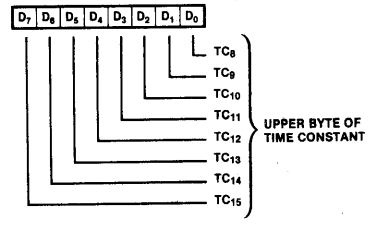
Read Register 10



Read Register 12



Read Register 13



Read Register 15

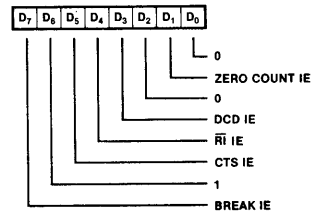


Figure 8. Read Register Bit Functions

Programming (Continued)

Write Registers. The Z-ASCC contains 11 write registers (12 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 9 shows the format of each write register.

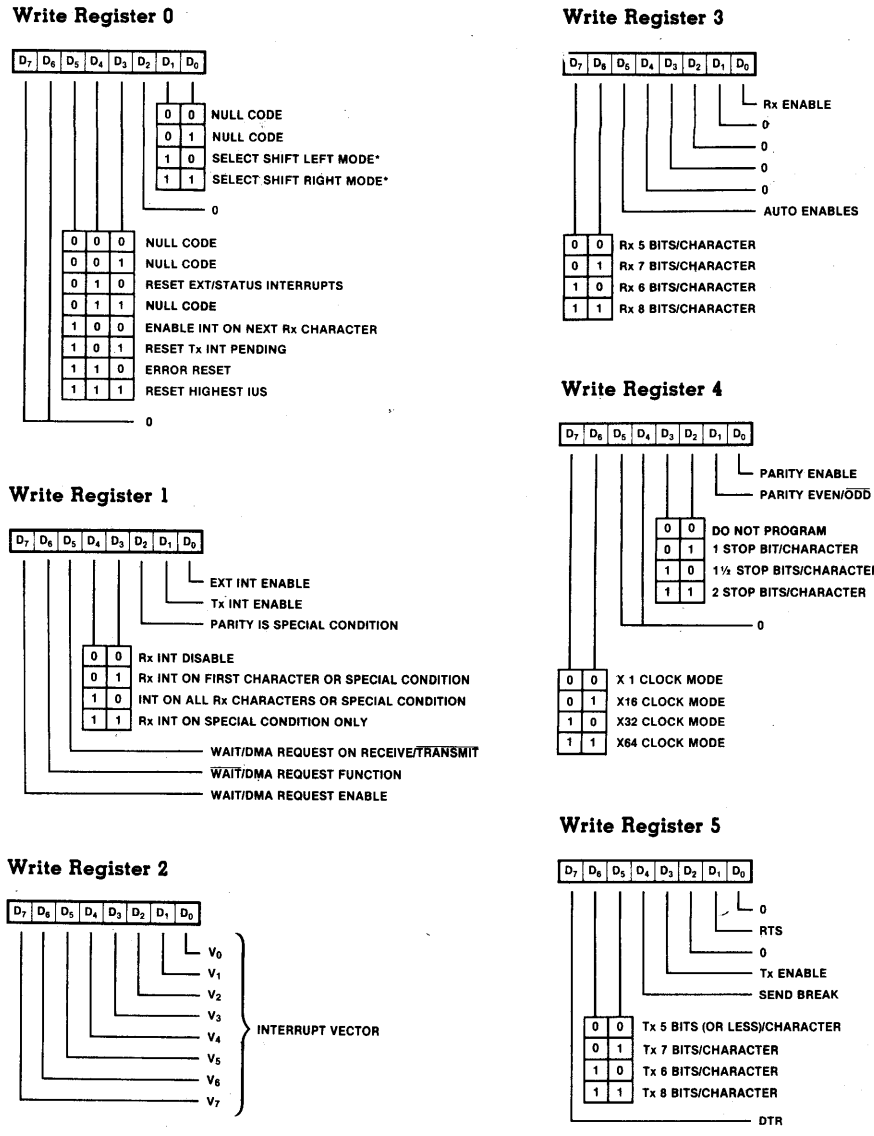
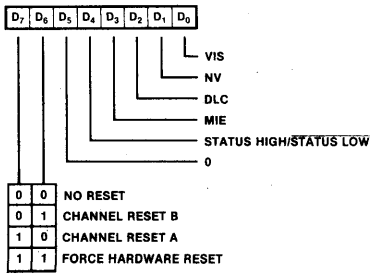


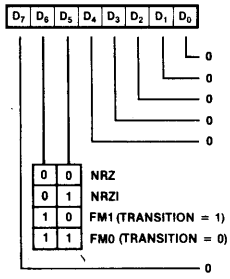
Figure 9. Write Register Bit Functions

Programming (Continued)

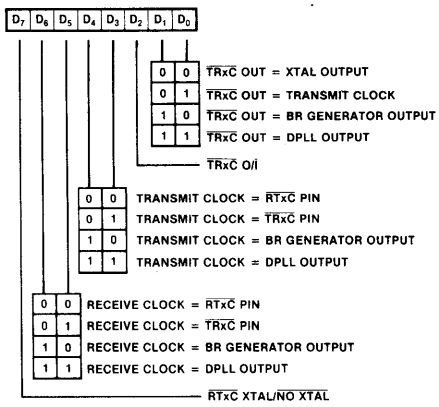
Write Register 9



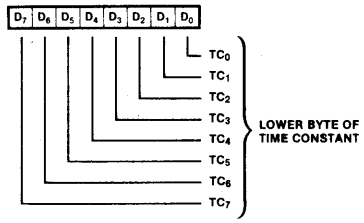
Writer Register 10



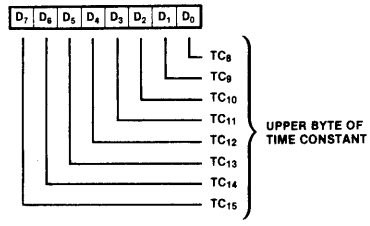
Write Register 11



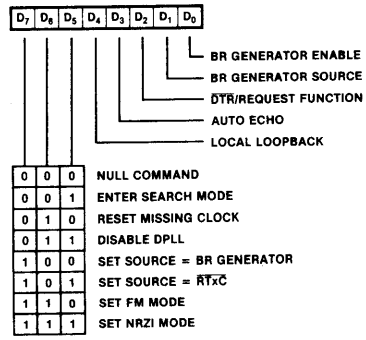
Write Register 12



Write Register 13



Write Register 14



Write Register 15

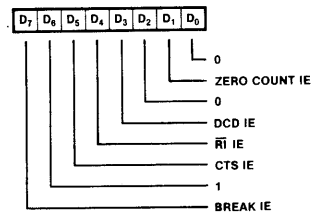


Figure 9. Write Register Bit Functions (Continued)

Timing

The Z-ASCC generates internal control signals from $\overline{AS}$ and $\overline{DS}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{AS}$ and $\overline{DS}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the Z-ASCC. The recovery time required for proper operation is specified from the rising edge of $\overline{DS}$ in the first transaction involving the Z-ASCC to the falling edge of $\overline{DS}$ in the second transaction involving the Z-ASCC. This time must be at least 6 PCLK cycles plus 200 ns.

Read Cycle Timing. Figure 10 illustrates Read cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be High to indicate a Read cycle. CS_1 must also be High for the Read cycle to occur. The data bus drivers in the Z-ASCC are then enabled while $\overline{DS}$ is Low.

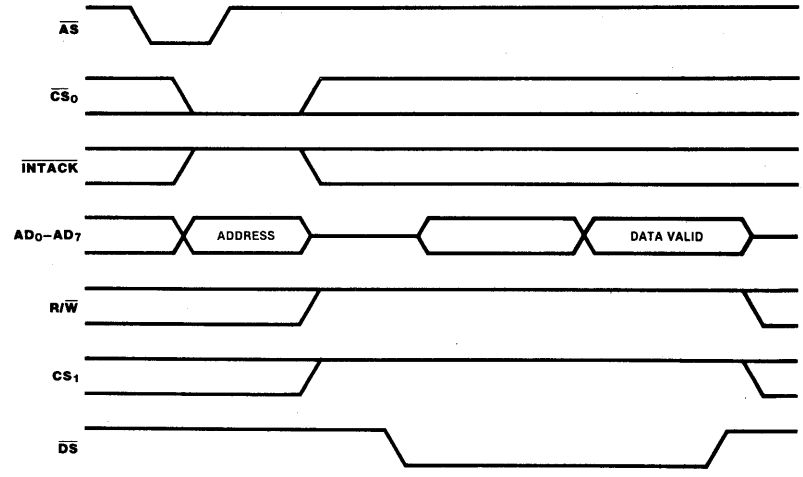


Figure 10. Read Cycle Timing

Write Cycle Timing. Figure 11 illustrates Write cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. R/$\overline{W}$ must be Low to indicate a Write cycle. CS_1 must be High for the Write cycle to occur. $\overline{DS}$ Low strobes the data into the Z-ASCC.

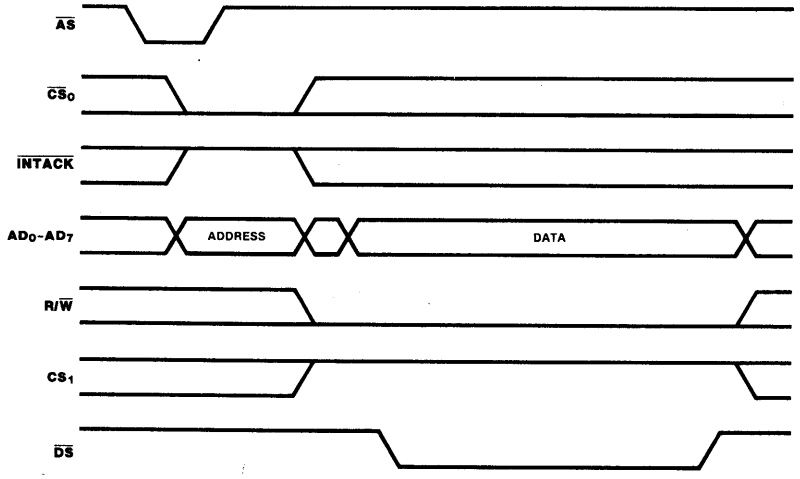


Figure 11. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 12 illustrates Interrupt Acknowledge cycle timing. The address on AD_0–AD_7 and the state of $\overline{CS_0}$ and $\overline{INTACK}$ are latched by the rising edge of $\overline{AS}$. However, if $\overline{INTACK}$ is Low, the address and $\overline{CS_0}$ are ignored. The state of the R/$\overline{W}$ and CS_1 are also ignored for the duration of the Interrupt Acknowledge

Timing (Continued)

cycle. Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the Z-ASCC and IEI is High when $\overline{DS}$ falls, the Acknowledge cycle was intended for the Z-ASCC. In this case, the Z-ASCC may be programmed to respond to $\overline{DS}$ Low by placing its interrupt vector on AD_0-AD_7. It then sets the appropriate Interrupt-Under-Service latch internally.

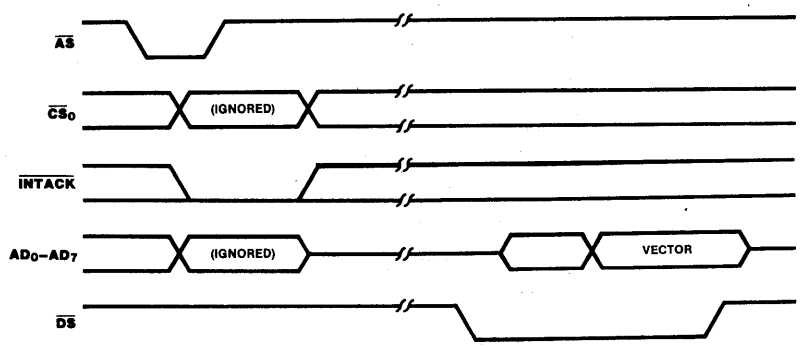


Figure 12. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

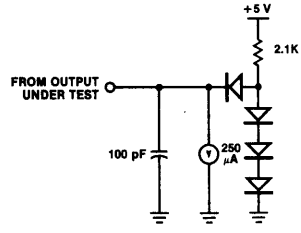


Figure 13. Standard Test Load

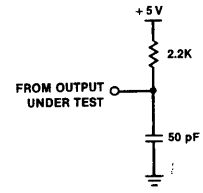


Figure 14. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4V$
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

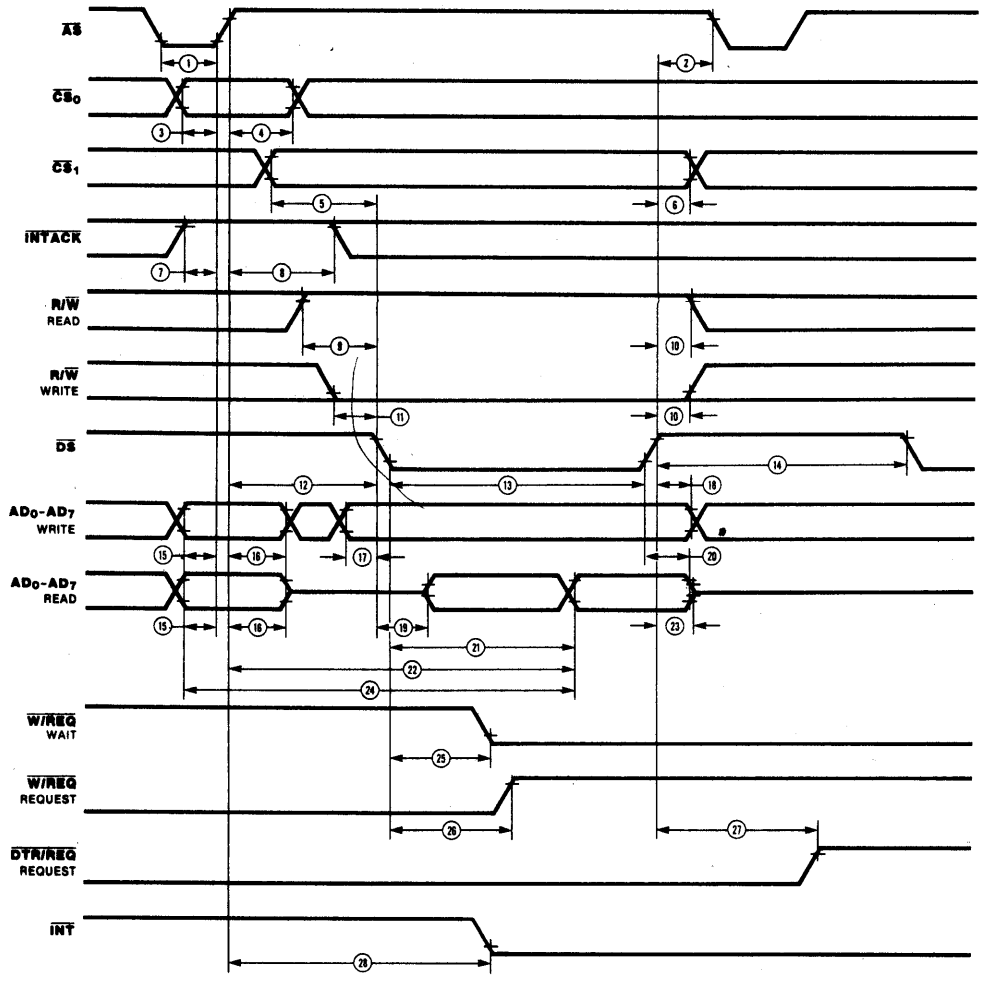
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins
C_{OUT}	Output Capacitance		15	pF	Returned to Ground
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

Read and Write Timing

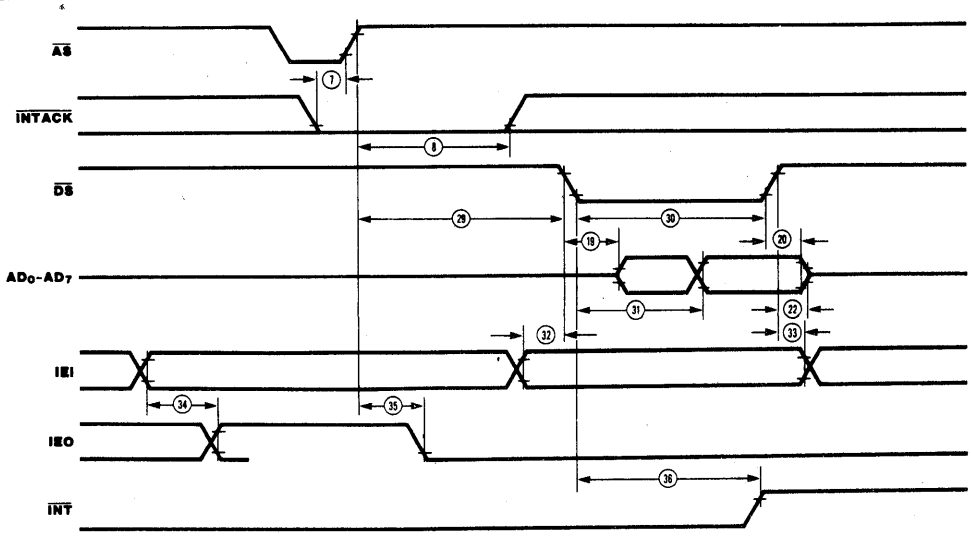


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwAS	$\overline{AS}$ Low Width	70		50		
2	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
3	TsCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
4	ThCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
5	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		80		1
6	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	55		40		1
7	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
8	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
10	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		
11	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
12	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		40		
13	TwDS1	$\overline{DS}$ Low Width	390		250		
14	TrC	Valid Access Recovery Time	6TcPC +200		6TcPC +130		2
15	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
16	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
17	TsDW(DS)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
18	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		
19	TdDS(DA)	$\overline{DS}$ ↓ to Data Active Delay	0		0		
20	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
21	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180	
22	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Valid Delay		520		335	

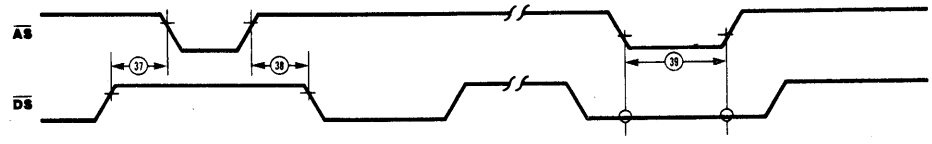
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Parameter applies only between transactions involving the Z-ASCC.

*Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

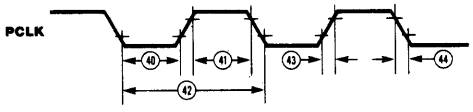
Interrupt Acknowledge Timing



Reset Timing



Cycle Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
23	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	3
24	TdA(DR)	Address Required Valid to Read Data Valid Delay		570		420	
25	TdDS(W)	$\overline{DS}$ ↓ to Wait Valid Delay		240		200	4
26	TdDSf(REQ)	$\overline{DS}$ ↓ to $\overline{W/REQ}$ Not Valid Delay		240		200	
27	TdDSr(REQ)	$\overline{DS}$ ↑ to $\overline{DTR/REQ}$ Not Valid Delay		5TcPC +300		5TcPC +250	
28	TdAS(INT)	$\overline{AS}$ ↑ to $\overline{INT}$ Valid Delay		500		500	4
29	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (Acknowledge) Delay					5
30	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	390		250		
31	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		250		180	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	120		100		
33	ThIEI(DSA)	IEI to $\overline{DS}$ ↑ (Acknowledge) Hold Time	0		0		
34	TdIEI(IEO)	IEI to IEO Delay		120		100	
35	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		250		250	6
36	TdDSA(INT)	$\overline{DS}$ ↓ (Acknowledge) to $\overline{INT}$ Inactive Delay		500		500	4
37	TdDS(ASQ)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay for No Reset	30		15		
38	TdASQ(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay for No Reset	30		30		
39	TwRES	$\overline{AS}$ and $\overline{DS}$ Coincident Low for Reset	250		250		7
40	TwPCl	PCLK Low Width	105	2000	70	1000	
41	TwPCh	PCLK High Width	105	2000	70	1000	
42	TcPC	PCLK Cycle Time	250	4000	165	2000	
43	TrPC	PCLK Rise Time		20		15	
44	TfPC	PCLK Fall Time		20		10	

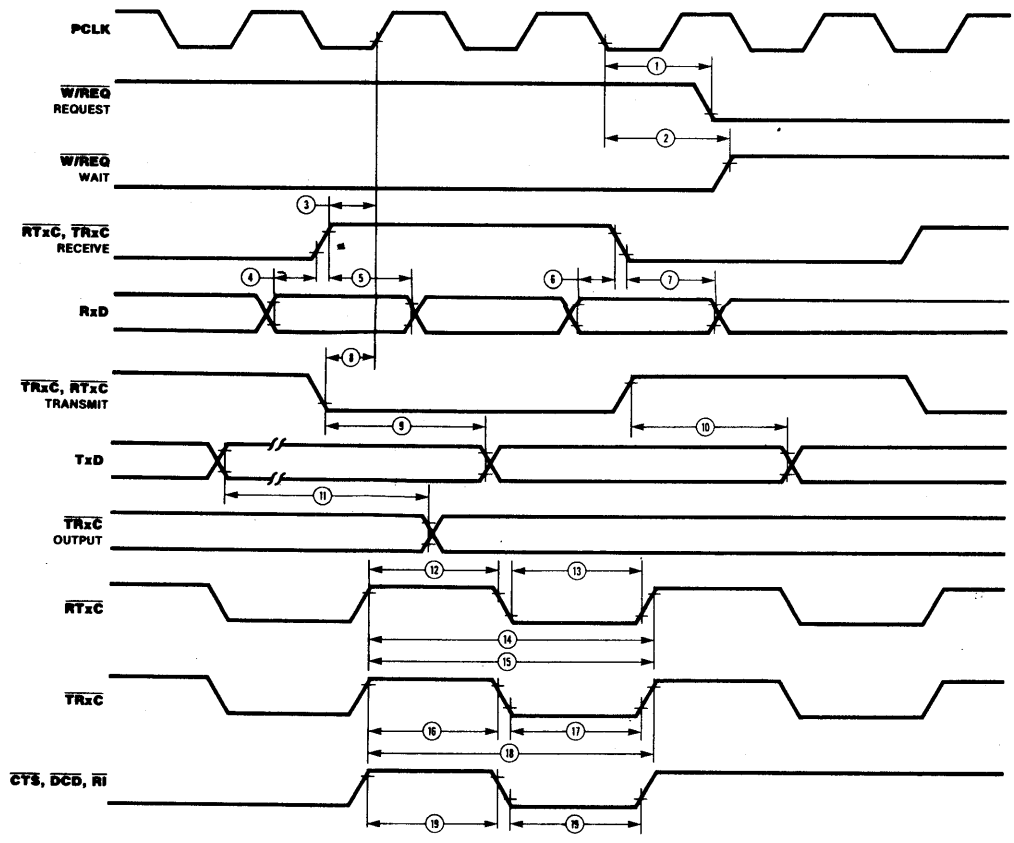
NOTES:

3. Float delay is defined as the time required for a ±0.5 V change in the output with a maximum dc load and minimum ac load.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any Z-ASCC in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the highest priority device in the daisy chain, TsIEI(DSA) for the Z-ASCC, and TdIEIf(IEO) for each device separating them in the daisy chain.
6. Parameter applies only to a Z-ASCC pulling $\overline{INT}$ Low at the beginning of the Interrupt Acknowledge transaction.
7. Internal circuitry allows for the reset provided by the Z8 to be recognized as a reset by the Z-ASCC.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Units in nanoseconds (ns).

General Timing

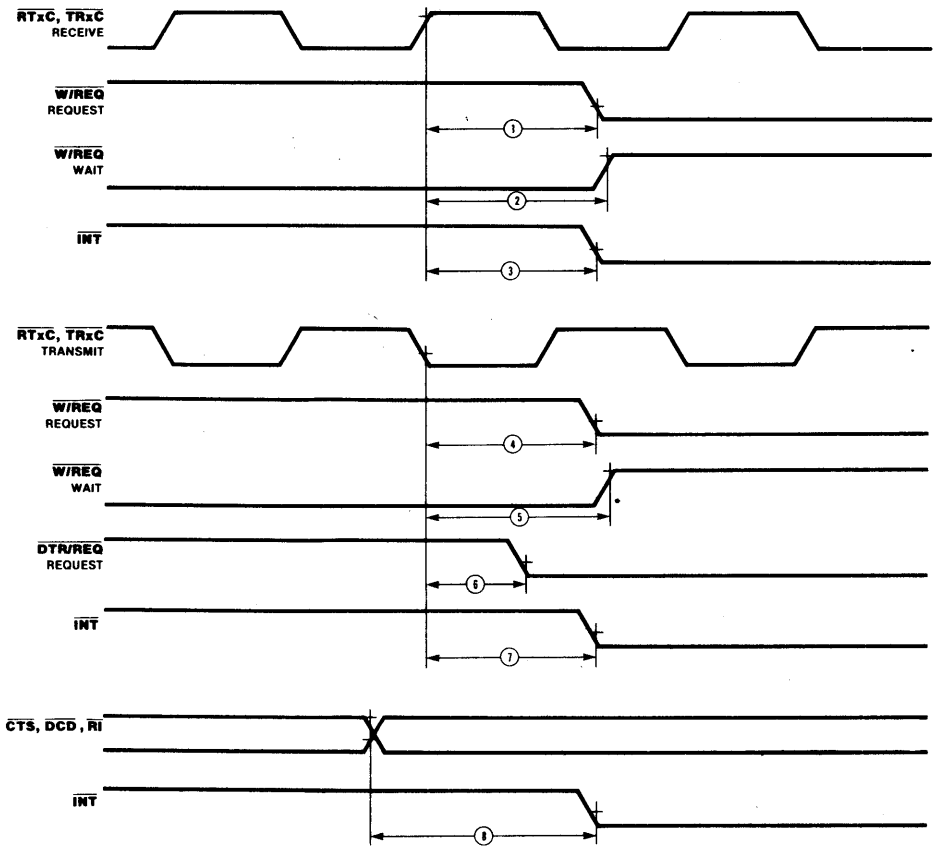


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdPC(REQ)	PCLK ↓ to W/REQ Valid		250		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350		350	
3	TsRXC(PC)	RxC ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	80	TwPCl	70	TwPCl	1,4
4	TsRXD(RXCr)	RxD to RxC ↑ Setup Time (X1 Mode)	0		0		1
5	ThRXD(RXCr)	RxD to RxC ↑ Hold Time (X1 Mode)	150		150		1
6	TsRXD(RXCf)	RxD to RxC ↓ Setup Time (X1 Mode)	0		0		1,5
7	ThRXD(RXCf)	RxD to RxC ↓ Hold Time (X1 Mode)	150		150		1,5
8	TsTXC(PC)	TxC ↓ to PCLK ↑ Setup Time	0		0		2,4
9	TdTXCf(TXD)	TxC ↓ to TxD Delay (X1 Mode)		300		300	2
10	TdTXCr(TXD)	TxC ↑ to TxD Delay (X1 Mode)		300		300	2,5
11	TdTXD(TRX)	TxD to TRxC Delay (Send Clock Echo)					
12	TwRTXh	RTxC High Width	180		180		6
13	TwRTXl	RTxC Low Width	180		180		6
14	TcRTX	RTxC Cycle Time	400		400		6
15	TcRTXX	Crystal Oscillator Period	250	1000	250	1000	3
16	TwTRXh	TRxC High Width	180		180		6
17	TwTRXl	TRxC Low Width	180		180		6
18	TcTRX	TRxC Cycle Time	400		400		6
19	TwEXT	DCD or CTS or RI Pulse Width	200		200		

NOTES:
1. RxC is RTxC or TRxC, whichever is supplying the receive clock.
2. TxC is TRxC or RTxC, whichever is supplying the transmit clock.
3. Both RTxC and RI have 30 pF capacitors to the ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between RxC and PCLK or TxC and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to chip PCLK requirements.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

System Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	8	12	8	12	2,4
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	12	8	12	1,2,4
3	TdRXC(INT)	$\overline{RxC}$ ↑ $\overline{INT}$ Valid Delay	8 +2	12 +3	8 +2	12 +3	1,2,4 5
4	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	5	8	5	8	3,4
5	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	8	5	8	1,3,4
6	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DTR/REQ}$ Valid Delay	4	7	4	7	3,4
7	TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	4 +2	6 +3	4 +2	6 +3	1,3,4 5
8	TdEXT(INT)	$\overline{DCD}$, $\overline{RI}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	3	2	3	1,5

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
4. Units equal to $\overline{TcPC}$.
5. Units equal to $\overline{AS}$.

* Timings are preliminary and subject to change.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8031	CE	4.0 MHz	Z-ASCC (40-pin)	Z8031A	CE	6.0 MHz	Z-ASCC (40-pin)
Z8031	CM	4.0 MHz	Same as above	Z8031A	CM	6.0 MHz	Same as above
Z8031	CMB	4.0 MHz	Same as above	Z8031A	CMB	6.0 MHz	Same as above
Z8031	CS	4.0 MHz	Same as above	Z8031A	CS	6.0 MHz	Same as above
Z8031	DE	4.0 MHz	Same as above	Z8031A	DE	6.0 MHz	Same as above
Z8031	DS	4.0 MHz	Same as above	Z8031A	DS	6.0 MHz	Same as above
Z8031	PE	4.0 MHz	Same as above	Z8031A	PE	6.0 MHz	Same as above
Z8031	PS	4.0 MHz	Same as above	Z8031A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to 125°C, MB = -55°C to +125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8036 Z8000™ Z-CIO Counter/Timer and Parallel I/O Unit

Zilog

Product Specification

September 1983

Features

- Two independent 8-bit, double-buffered, bidirectional I/O ports plus a 4-bit special-purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/$\overline{\text{WAIT}}$ signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or nonretriggerable.
- Easy to use since all registers are read/write and directly addressable.

General Description

The Z8036 Z-CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications.

The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique address so that it can be accessed directly—no special sequential operations are required. The Z-CIO is directly Z-Bus compatible.

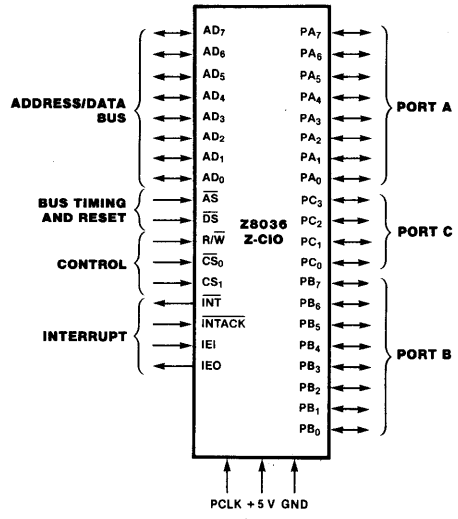


Figure 1. Pin Functions

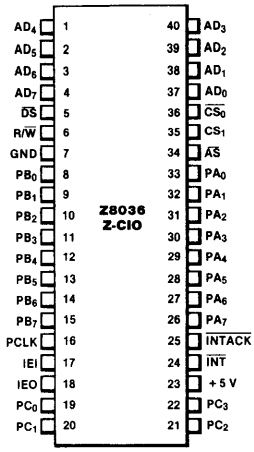


Figure 2. Pin Assignments

Pin Description

AD_0–AD_7. *Z-Bus Address/Data lines* (bidirectional/3-state). These multiplexed Address/Data lines are used for transfers between the CPU and Z-CIO.

$\overline{AS}$*. *Address Strobe* (input, active Low). Addresses, $\overline{INTACK}$, and $\overline{CS_0}$ are sampled while $\overline{AS}$ is Low.

$\overline{CS_0}$ and CS_1. *Chip Select 0* (input, active Low) *and Chip Select 1* (input, active High). $\overline{CS_0}$ and CS_1 must be Low and High, respectively, in order to select a device. $\overline{CS_0}$ is latched by $\overline{AS}$.

$\overline{DS}$*. *Data Strobe* (input, active Low). $\overline{DS}$ provides timing for the transfer of data into or out of the Z-CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting Z-CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

*When $\overline{AS}$ and $\overline{DS}$ are detected Low at the same time (normally an illegal condition), the Z-CIO is reset.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the Z-CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates to the Z-CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ is sampled while $\overline{AS}$ is Low.

PA_0–PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the Z-CIO's Port A and external devices.

PB_0–PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the Z-CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0–PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the Z-CIO's Port C.

PCLK. *(input, TTL-compatible).* This is a peripheral clock that may be, but is not necessarily, the CPU clock. It is used with timers and REQUEST/$\overline{WAIT}$ logic.

R/$\overline{W}$. *Read/Write* (input). R/$\overline{W}$ indicates that the CPU is reading from (High) or writing to (Low) the Z-CIO.

Architecture

The Z8036 Z-CIO Counter/Timer and Parallel I/O element (Figure 3) consists of a Z-Bus interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose

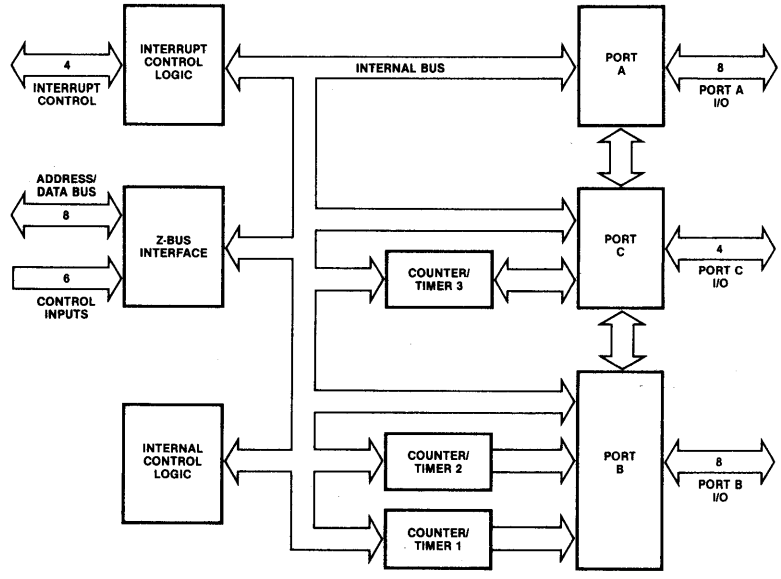


Figure 3. Z-CIO Block Diagram

Architecture (Continued)

4-bit port), three 16-bit counter/timers, an interrupt control logic block, and the internal control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

The two general-purpose 8-bit I/O ports (Figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers: the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

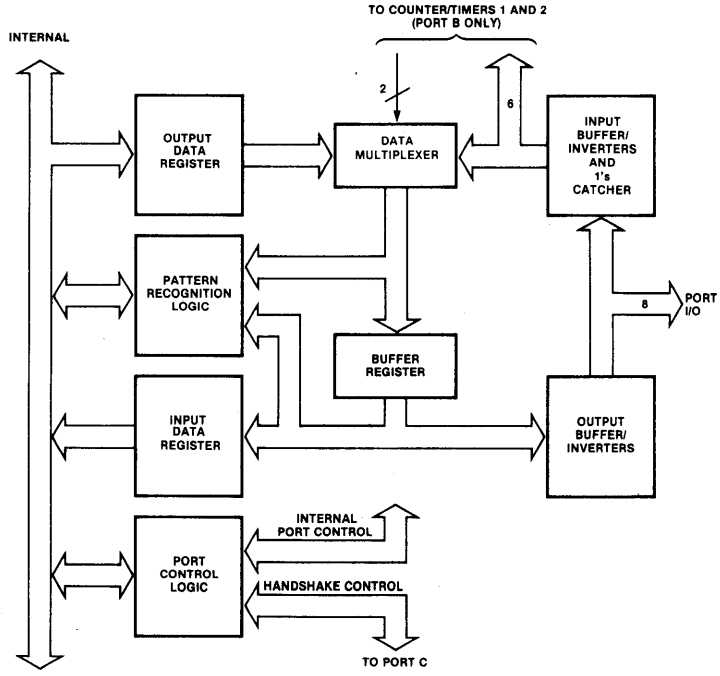


Figure 4. Ports A and B Block Diagram

Architecture (Continued)

The function of the special-purpose 4-bit port, Port C (Figure 5), depends upon the roles of Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path: the Data Path Polarity, Data Direction, and Special I/O Control registers.

The three counter/timers (Figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Counter register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer: counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or nonretriggerable. With these and other options, most counter/timer applications are covered.

The interrupt control logic provides standard Z-Bus interrupt capabilities. There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

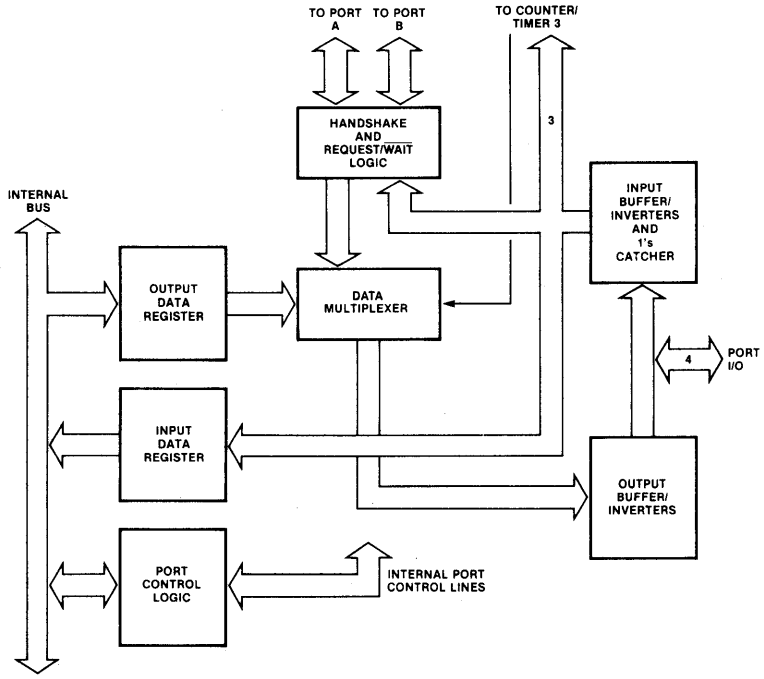


Figure 5. Port C Block Diagram

Architecture (Continued)

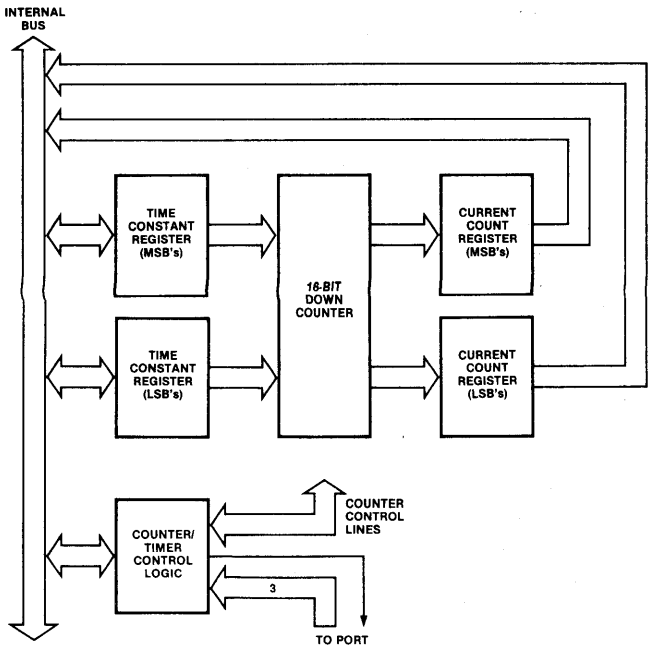


Figure 6. Counter/Timer Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the Z-CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/$\overline{\text{WAIT}}$ line can also be provided so that Z-CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/$\overline{\text{WAIT}}$ can be used as input or output bits (individually data direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to a 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Functional Description (Continued)

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The Z-CIO provides four different handshakes for its ports: Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in Table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3.

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked, Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (Table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the Z-CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the Z-CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another Z-CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{DAV}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{DAV}$ is

Port A/B Configuration	PC_3	PC_2	PC_1	PC_0
Ports A and B: Bit Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O
Port B: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O	RFD or $\overline{DAV}$	$\overline{ACKIN}$
Port A or B: Input Port (3-Wire Handshake)	RFD (Output)	$\overline{DAV}$ (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	DAC (Output)
Port A or B: Output Port (3-Wire Handshake)	$\overline{DAV}$ (Output)	DAC (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	RFD (Input)
Port A or B: Bidirectional Port (Interlocked or Strobed Handshake)	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	IN/$\overline{OUT}$

*Both Ports A and B can be specified input or output with Interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{WAIT}$.

Table 1. Port C Bit Utilization

Functional Description (Continued)

allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{DAV}$ goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input ($\overline{ACKIN}$) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked Handshake, the signal indicating the port is ready for another data transfer operates independently of the $\overline{ACKIN}$ input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (Figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (Figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the $\overline{ACKIN}$ path. The external $\overline{ACKIN}$ input triggers the timer and its output is used as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{DAV}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{DAV}$ output goes Low and the timer output is used as the actual $\overline{DAV}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

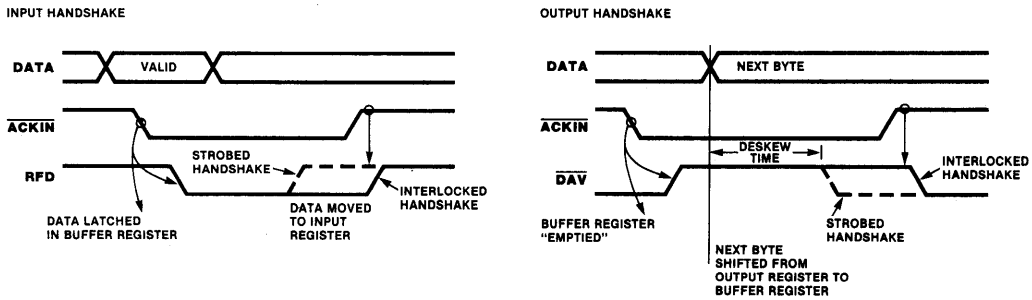


Figure 7. Interlocked and Strobed Handshakes

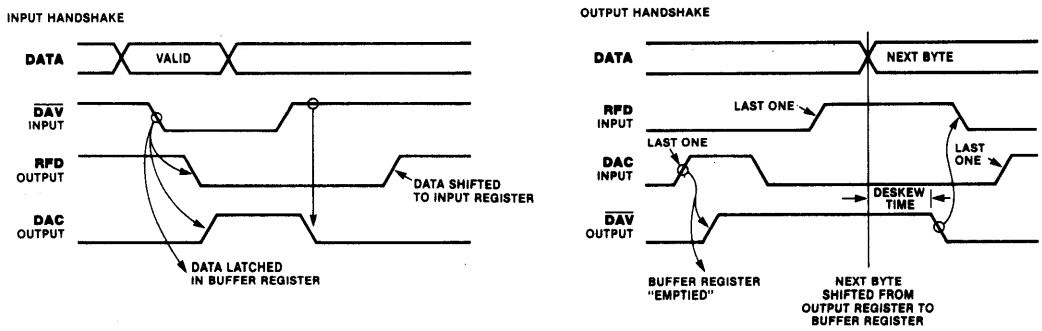


Figure 8. 3-Wire Handshake

Functional Description (Continued)

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the Z-Bus. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the Z-CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all nonmasked bit specifications in the AND mode or the satisfaction of any nonmasked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

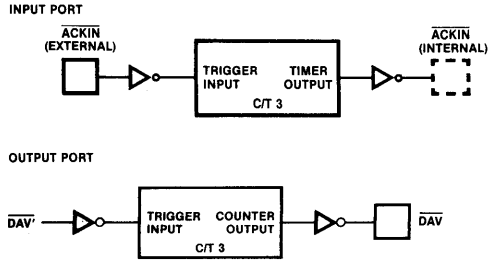


Figure 9. Pulsed Handshake

The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt. Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes: Transparent and Latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1, after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. Once the Acknowledge cycle is initiated, the vector is

Functional Description (Continued)

frozen until the corresponding IP is cleared. Where inputs that cause interrupts might change before the interrupt is serviced, the 1's catcher can be used to hold the value. Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register to the Input register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern-match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type applications when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

Table 2. Counter/Timer External Access

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four least significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (Table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output: pulse, one-shot, and square-wave. Figure 10 shows the counter/

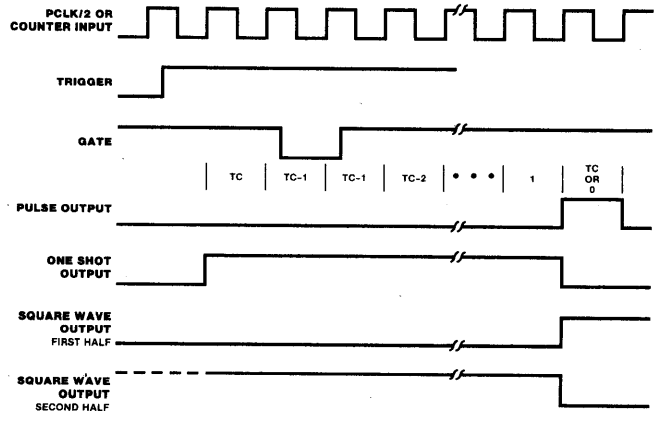


Figure 10. Counter/Timer Waveforms

Functional Description (Continued)

timer waveforms. When the Pulse mode is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal countdown sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/$\overline{\text{Single}}$ $\overline{\text{Cycle}}$ (C/$\overline{\text{SC}}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{\text{SC}}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{\text{SC}}$ bit is 1 each time the countdown counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65,536. The down-counter is loaded in one of three ways: by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal output if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Once the down-counter is loaded, the countdown sequence continues toward terminal count as long as all the counter/timers' hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a countdown sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the countdown sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to a 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways: by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The interrupts generated by the Z-CIO follow the Z-Bus operation as described more fully in the Zilog *Z-Bus Summary*. The Z-CIO has five potential sources of interrupts: the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order: Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The Z-CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device,

Functional Description (Continued)

the Z-CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the Z-CIO contains three bits for the control and status of the interrupt logic: an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS) status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{INT}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. It can also be set directly by the CPU. Its primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, an interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service; however, the $\overline{INT}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the Z-CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the Z-CIO's IEO is forced Low, independent of the state of the Z-CIO or its IEI input, and all lower priority devices' interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the Z-CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The Z-CIO contains three vector registers: one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1. The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the Z-CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

Programming

Programming the Z-CIO entails loading control registers with bits to implement the desired operation. Individual enable bits are provided for the various major blocks so that erroneous operations do not occur while the part is being initialized. Before the ports are enabled, IPs cannot be set, REQUEST and $\overline{WAIT}$ cannot be asserted, and all outputs remain high-impedance. The handshake lines are ignored until Port C is enabled. The counter/timers cannot be triggered until their enable bits are set.

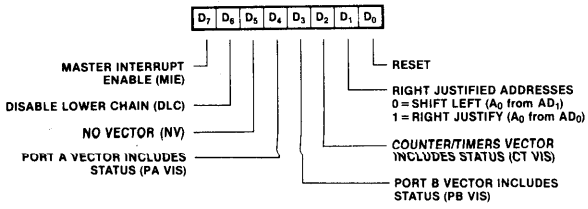
The Z-CIO is reset by forcing $\overline{AS}$ and $\overline{DS}$ Low simultaneously or by writing a 1 to the Reset bit. Once reset, the only thing that can be done is to read and write the Reset bit. Writes to all other bits are ignored and all reads return 0s. In this state, all control bits are forced to 0. Only after clearing the Reset bit (by writing to it) can the other command bits be programmed.

Register Addressing. The Z-CIO allows two schemes for register addressing. Both schemes use only six of the eight bits of the address/data bus. The scheme used is determined by the Right Justify Address (RJA) bit in the Master Interrupt Control register. When RJA equals 0, address bus bits 0 and 7 are ignored, and bits 1 through 6 are decoded for the register address (A_0 from AD_1). When RJA equals 1, bits 0 through 5 are decoded for the register address (A_0 from AD_0). In the following register descriptions, only six bits are shown for addresses and represent address/data bus bits 0 through 5 or 1 through 6, depending on the state of the RJA bit.

Registers

Master Interrupt Control Register

Address: 000000
(Read/Write)



Master Configuration Control Register

Address: 000001
(Read/Write)

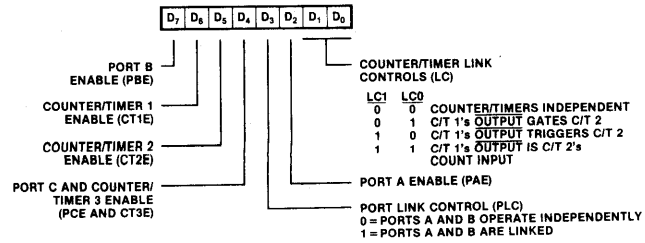
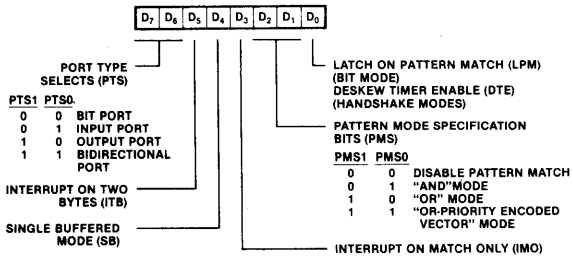


Figure 11. Master Control Registers

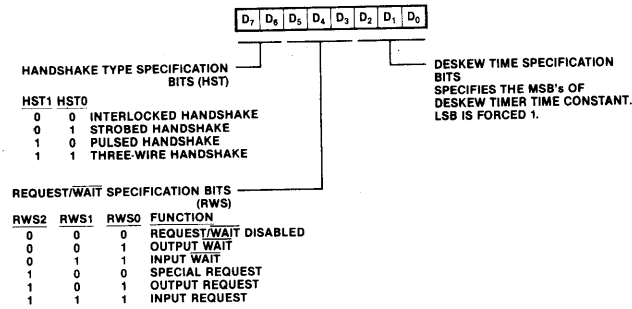
Port Mode Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)



Port Handshake Specification Registers

Addresses: 100001 Port A
101001 Port B
(Read/Write)



Port Command and Status Registers

Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)

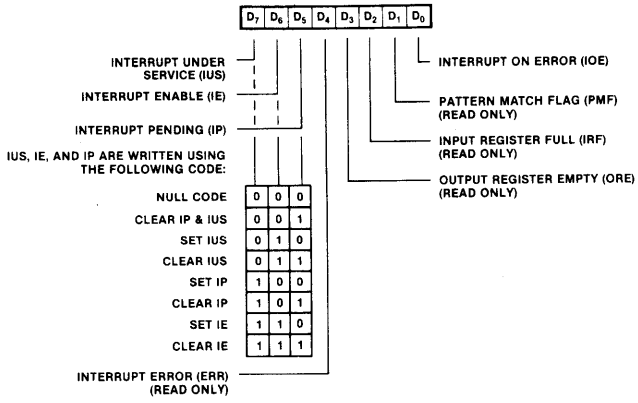
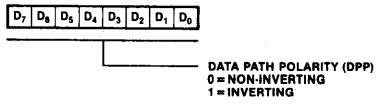


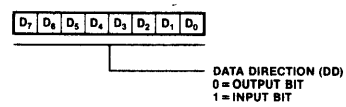
Figure 12. Port Specification Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

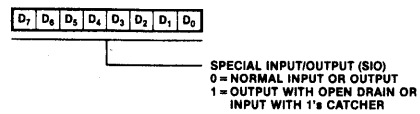
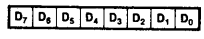


Figure 13. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A
001110 Port B
(Read/Write)



Port C Data Register
Address: 001111
(Read/Write)

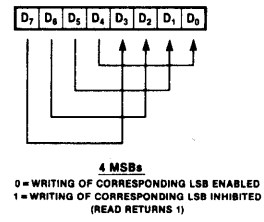
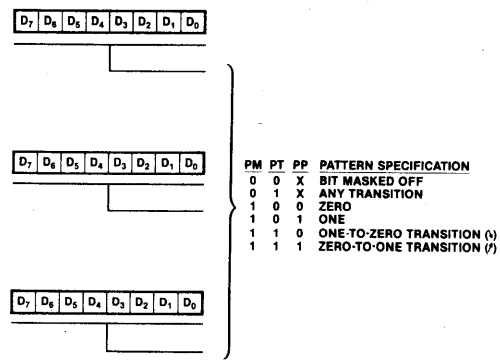


Figure 14. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)



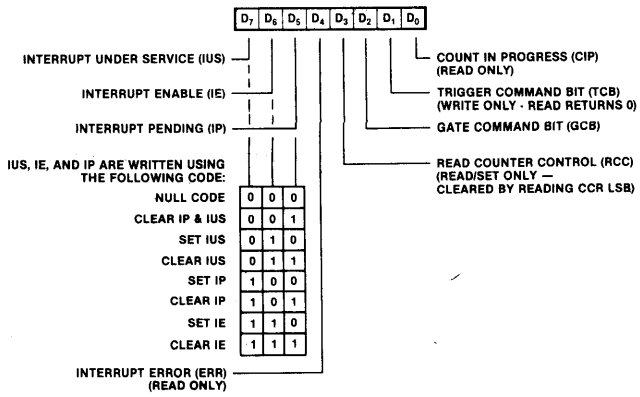
PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE-TO-ZERO TRANSITION (\)
1	1	1	ZERO-TO-ONE TRANSITION (/)

Figure 15. Pattern Definition Registers

Registers
(Continued)

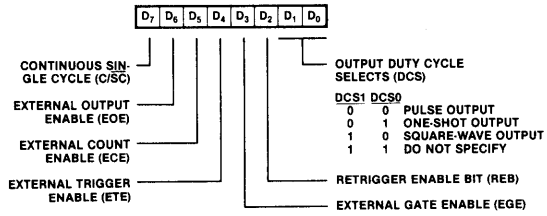
Counter/Timer Command and Status Registers

Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)



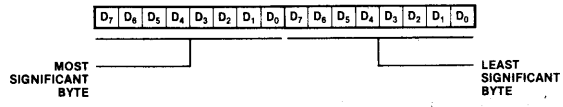
Counter/Timer Mode Specification Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)



Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

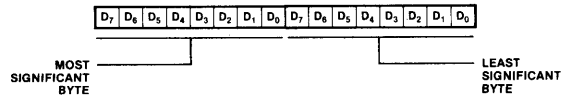
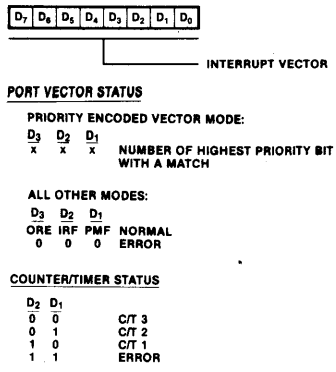


Figure 16. Conter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



Current Vector Register
Address: 011111
(Read Only)

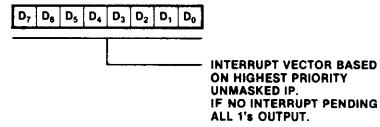


Figure 17. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address*	Register Name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Most Often Accessed Registers

Address*	Register Name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data
001110	Port B's Data
001111	Port C's Data

Counter/Timer Related Registers

Address*	Register Name
010000	Counter/Timer 1's Current Count-MSBs
010001	Counter/Timer 1's Current Count-LSBs
010010	Counter/Timer 2's Current Count-MSBs
010011	Counter/Timer 2's Current Count-LSBs
010100	Counter/Timer 3's Current Count-MSBs
010101	Counter/Timer 3's Current Count-LSBs
010110	Counter/Timer 1's Time Constant-MSBs
010111	Counter/Timer 1's Time Constant-LSBs
011000	Counter/Timer 2's Time Constant-MSBs
011001	Counter/Timer 2's Time Constant-LSBs
011010	Counter/Timer 3's Time Constant-MSBs
011011	Counter/Timer 3's Time Constant-LSBs
011100	Counter/Timer 1's Mode Specification
011101	Counter/Timer 2's Mode Specification
011110	Counter/Timer 3's Mode Specification
011111	Current Vector

Port A Specification Registers

Address*	Register Name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Port B Specification Registers

Address*	Register Name
101000	Port B's Mode Specification
101001	Port B's Handshake Specification
101010	Port B's Data Path Polarity
101011	Port B's Data Direction
101100	Port B's Special I/O Control
101101	Port B's Pattern Polarity
101110	Port B's Pattern Transition
101111	Port B's Pattern Mask

*When RJA = 0, A_0 from AD_1; when RJA = 1, A_0 from AD_0

Timing

Read Cycle. The CPU places an address on the address/data bus. The more significant bits and status information are combined and decoded by external logic to provide two Chip Selects ($\overline{CS}_0$ and CS_1). Six bits of the least significant byte of the address are latched within the Z-CIO and used to specify a Z-CIO register. The data from the register specified is strobed onto the address/data bus when the CPU issues a Data Strobe ($\overline{DS}$). If the register indicated by the address does not exist, the Z-CIO remains high-impedance.

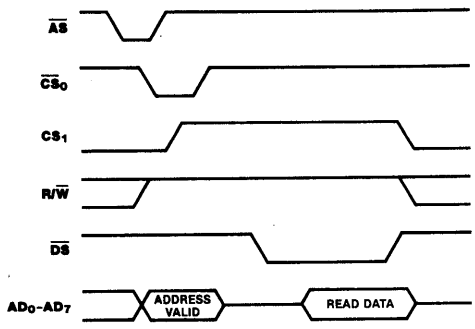


Figure 18. Read Cycle Timing

Write Cycle. The CPU places an address on the address/data bus. The more significant bits and status information are combined and decoded by external logic to provide two Chip Selects ($\overline{CS}_0$ and CS_1). Six bits of the least significant byte of the address are latched within the Z-CIO and used to specify a Z-CIO register. The CPU places the data on the address/data bus and strobes it into the Z-CIO register by issuing a Data Strobe ($\overline{DS}$).

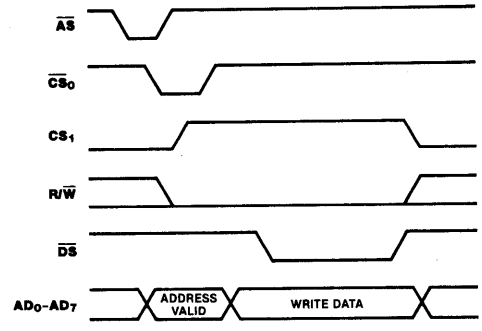
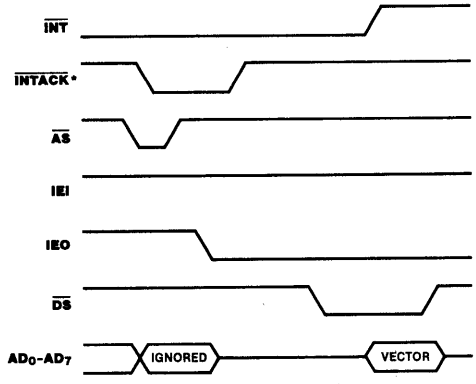


Figure 19. Write Cycle Timing

Interrupt Acknowledge Cycle. When one of the IP bits in the Z-CIO goes High and interrupts are enabled, the Z-CIO pulls its $\overline{INT}$ output line Low, requesting an interrupt. The CPU responds with an Interrupt Acknowledge cycle. When $\overline{INTACK}$ goes Low with IP set, the Z-CIO pulls its Interrupt Enable Out (IEO) Low, disabling all lower priority devices on the daisy chain. The CPU reads the Z-CIO interrupt vector by issuing a Low $\overline{DS}$, thereby strobing the interrupt vector onto the address/data bus. The IUS that corresponds to the IP is also set, which causes IEO to remain Low.



*$\overline{INTACK}$ is decoded from Z8000 status.

Figure 20. Interrupt Acknowledge Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

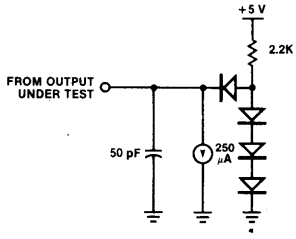


Figure 21. Standard Test Load

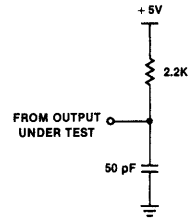


Figure 22. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
			0.5	V	$I_{OL} = +3.2$ mA
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		200	mA	

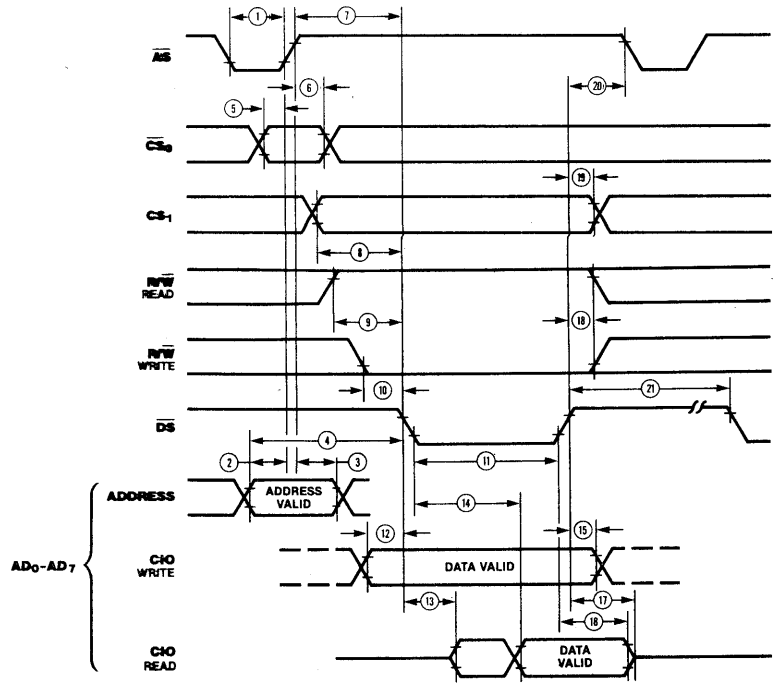
$V_{CC} = 5\ V \pm 5\%$ unless otherwise specified, over specified temperature range.

Capacitance

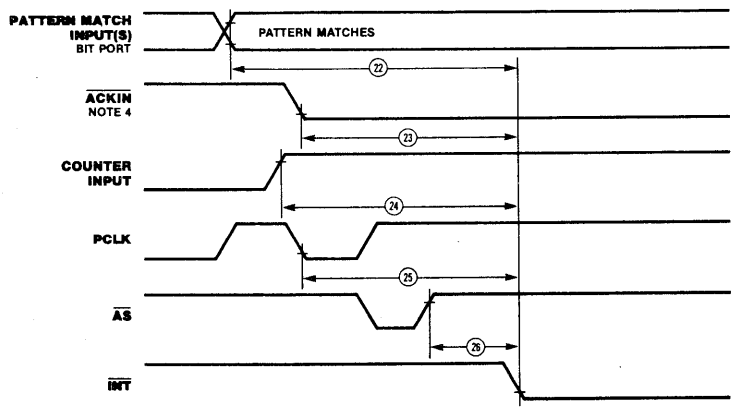
Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

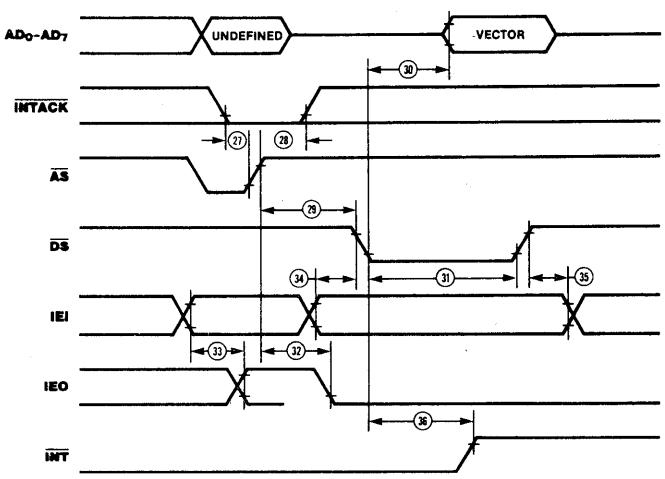
CPU Interface Timing



Interrupt Timing



Interrupt Acknowledge Timing

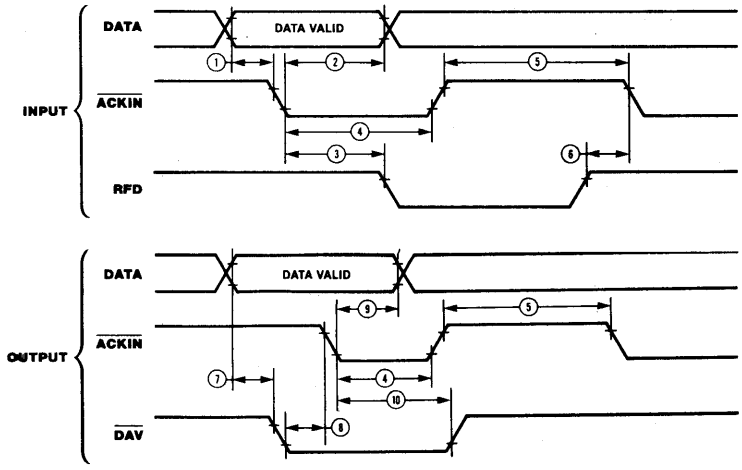


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwAS	$\overline{AS}$ Low Width	70	2000	50	2000	
2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
4	TsA(DS)	Address to $\overline{DS}$ ↓ Setup Time	130		100		1
5	TsCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
6	ThCSO(AS)	$\overline{CS_0}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
7	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60		40		1
8	TsCS1(DS)	CS_1 to $\overline{DS}$ ↓ Setup Time	100		80		
9	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
10	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
11	TwDS	$\overline{DS}$ Low Width	390		250		
12	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
13	TdDS(DRV)	$\overline{DS}$ (Read) ↓ to Address Data Bus Driven	0		0		
14	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180	
15	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		
16	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
17	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	2
18	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		
19	ThCS1(DS)	CS_1 to $\overline{DS}$ ↑ Hold Time	55		40		
20	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
21	Trc	Valid Access Recovery Time	1000		650		3
22	TdPM(INT)	Pattern Match to $\overline{INT}$ Delay (Bit Port)		1+800		1+800	6
23	TdACK(INT)	$\overline{ACKIN}$ to $\overline{INT}$ Delay (Port with Handshake)		4+600		4+600	4,6
24	TdCI(INT)	Counter Input to $\overline{INT}$ Delay (Counter Mode)		1+700		1+700	6
25	TdPC(INT)	PCLK to $\overline{INT}$ Delay (Timer Mode)		1+700		1+700	6
26	TdAS(INT)	$\overline{AS}$ to $\overline{INT}$ Delay		300			
27	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
28	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
29	TsAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Setup Time	350		250		5
30	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		250		180	
31	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	390		250		
32	TdAS(IEO)	$\overline{AS}$ ↑ to IEO ↓ Delay ($\overline{INTACK}$ Cycle)		350		250	5
33	TdIEI(IEO)	IEI to IEO Delay		150		100	5
34	TsIEI(DSA)	IEO to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		70		5
35	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↑ Hold Time	100		70		
36	TdDSA(INT)	$\overline{DS}$ (Acknowledge) ↓ to $\overline{INT}$ ↑ Delay		600		600	

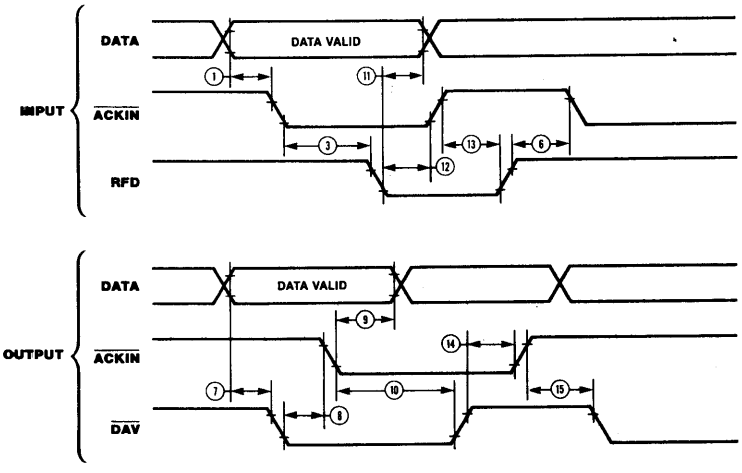
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{DS}$ ↑ of one CIO access to $\overline{DS}$ ↓ of another CIO access.
4. The delay is from DAV ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake. One additional AS cycle is required for ports in the Single Buffered mode.
5. The parameters for the devices in any particular daisy chain must meet the following constraint: the delay from $\overline{AS}$ ↑ to $\overline{DS}$ ↓ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.
6. Units equal to AS cycle + ns.

* Timings are preliminary and subject to change.
† Units in nanoseconds(ns), except as noted.

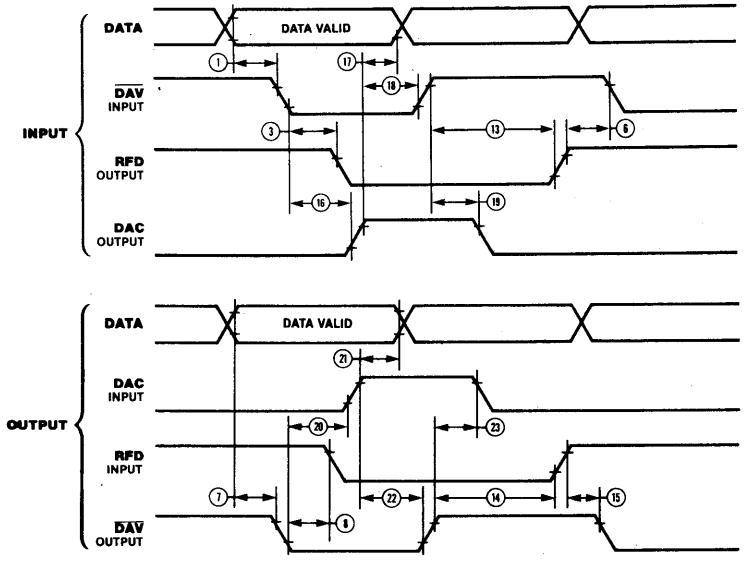
Strobed Handshake



Interlocked Handshake



3-Wire Handshake

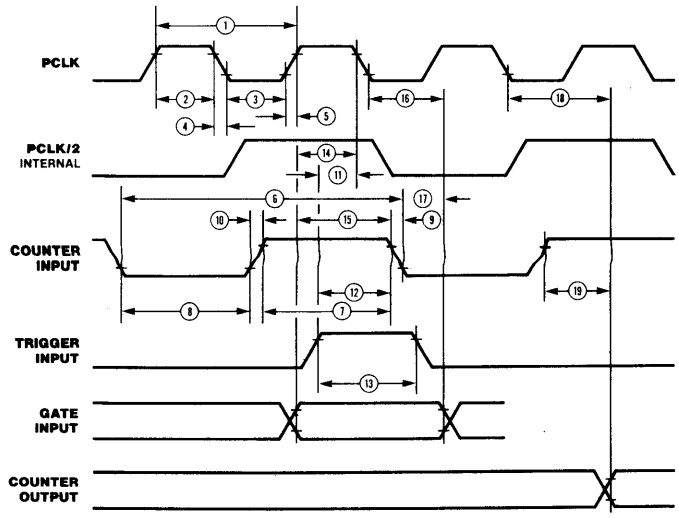


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Setup Time	0		0		
2	ThDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Hold Time - Strobed Handshake	500				
3	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		0		
4	TwACKl	$\overline{ACKIN}$ Low Width - Strobed Handshake	250				
5	TwACKh	$\overline{ACKIN}$ High Width - Strobed Handshake	250				
6	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		0		
7	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	25		20		1
8	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		0		
9	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↓ Hold Time	1		1		2
10	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	1		1		2
11	ThDI(RFD)	Data Input to RFD ↓ Hold Time - Interlocked Handshake	0		0		
12	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay - Interlocked Handshake	0		0		
13	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay - Interlocked and 3-Wire Handshake	0		0		
14	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑) - Interlocked and 3-Wire Handshake	0		0		
15	TdACK(DAV)	$\overline{ACKIN}$ ↑ (RFD ↑)to $\overline{DAV}$ ↓ Delay - Interlocked and 3-Wire Handshake	0		0		
16	TdDAVIf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay - Input 3-Wire Handshake	0		0		
17	ThDI(DAC)	Data Input to DAC ↑ Hold Time - 3-Wire Handshake	0		0		
18	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay - Input 3-Wire Handshake	0		0		
19	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay - Input 3-Wire Handshake	0		0		
20	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay - Output 3-Wire Handshake	0		0		
21	ThDO(DAC)	Data Output to DAC ↑ Hold Time - 3-Wire Handshake	1		1		2
22	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay - Output 3-Wire Handshake	1		1		2
23	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay - Output 3-Wire Handshake	0		0		

NOTES:
1. This time can be extended through the use of the deskew timers.
2. Units equal to $\overline{AS}$ cycle.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Counter/ Timer Timing

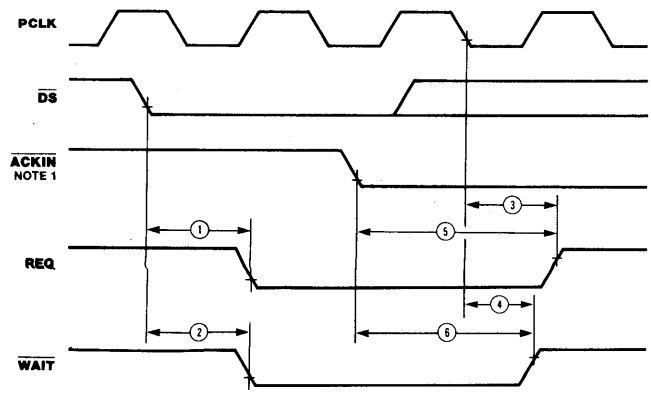


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TcPC	PCLK Cycle Time	250	4000	165	4000	1
2	TwPCh	PCLK High Width	105	2000	70	2000	
3	TwPCl	PCLK Low Width	105	2000	70	2000	
4	TfPC	PCLK Fall Time		20		10	
5	TrPC	PCLK Rise Time		20		15	
6	TcCI	Counter Input Cycle Time	500		330		
7	TCIh	Counter Input High Width	230		150		
8	TwCIl	Counter Input Low Width	230		150		
9	TfCI	Counter Input Fall Time		20		15	
10	TrCI	Counter Input Rise Time		20		15	
11	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (Timer Mode)	150				2
12	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (Counter Mode)	150				2
13	TwTI	Trigger Input Pulse Width (High or Low)	200				
14	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (Timer Mode)	100				2
15	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (Counter Mode)	100				2
16	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (Timer Mode)	100				2
17	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (Counter Mode)	100				2
18	TdPC(CO)	PCLK to Counter Output Delay (Timer Mode)		475			
19	TdCI(CO)	Counter Input to Counter Output Delay (Counter Mode)		475			

NOTES:
1. PCLK is only used with the counter/timers (in Timer mode), the deskew timers, and the REQUEST/$\overline{\text{WAIT}}$ logic. If these functions are not used, the PCLK input can be held low.
2. These parameters must be met to guarantee that trigger or gate are valid for the next counter/timer cycle.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

REQUEST/ WAIT Timing

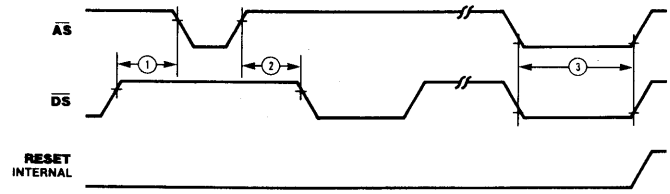


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDS(REQ)	$\overline{DS}$ ↓ to REQ ↓ Delay		500			
2	TdDS(WAIT)	$\overline{DS}$ ↓ to $\overline{WAIT}$ ↓ Delay		500			
3	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay		300			
4	TdPC(WAIT)	PCLK ↓ to $\overline{WAIT}$ ↑ Delay		300			
5	TdACK(REQ)	ACKIN ↓ to REQ ↑ Delay		3 + 2 + 1000			1,2
6	TdACK(WAIT)	ACKIN ↓ to $\overline{WAIT}$ ↑ Delay		10 + 600			3

NOTES:
1. The Delay is from DAV ↓ for the 3-Wire Input Handshake. The delay is from DAC ↑ for the 3-Wire Output Handshake.
2. Units equal to AS cycles + PCLK cycles + ns.
3. Units equal to PCLK cycles + ns.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Reset Timing

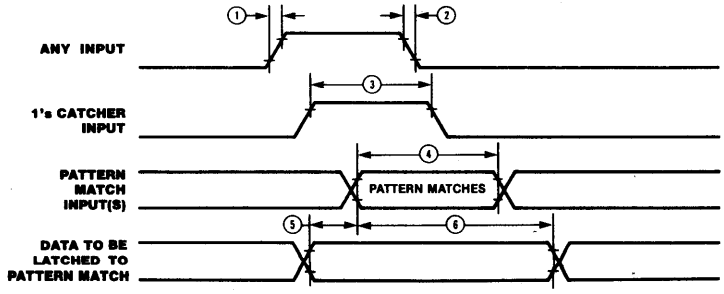


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for No Reset	40		15		
2	TdASQ(DS)	Delay from $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for No Reset	50		30		
3	TwRES	Minimum Width of $\overline{AS}$ and $\overline{DS}$ both Low for Reset	250		170		1

NOTES:
1. Internal circuitry allows for the reset provided by the Z8 ($\overline{DS}$ held Low while $\overline{AS}$ pulses) to be sufficient.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

Miscellaneous Port Timing

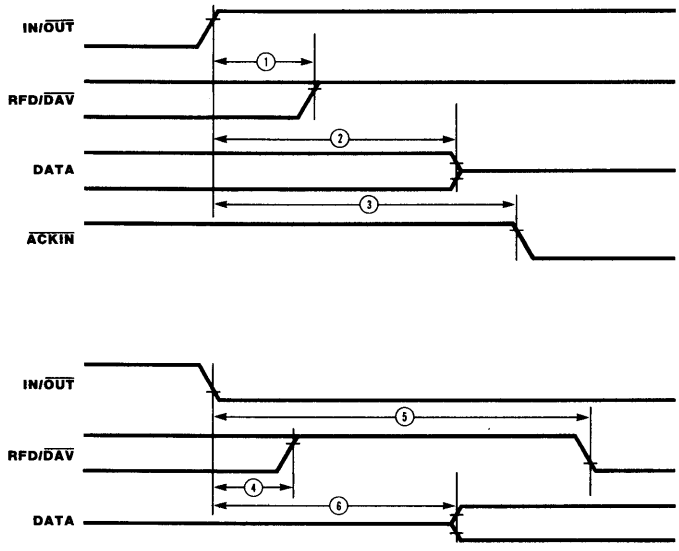


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TrI	Any Input Rise Time		100		100	
2	TfI	Any Input Fall Time		100		100	
3	Tw1's	1's Catcher High Width	250		170		1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		500		
5	TsPMD	Data Latched on Pattern Match Setup Time (Bit Port)	0		0		
6	ThPMD	Data Latched on Pattern Match Hold Time (Bit Port)	1000		650		

NOTES:
1. If the input is programmed inverting, a Low-going pulse of the same width will be detected.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

Bidirectional Port Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdIOr(DAV)	I/$\overline{O}$ ↑ to RFD/$\overline{DAV}$ High Delay		500		500	
2	TdIOr(DRZ)	I/$\overline{O}$ ↑ to Data Float Delay		500		500	
3	TdIOr(ACK)	I/$\overline{O}$ ↑ to $\overline{ACKIN}$ ↓ Delay					2
4	TdIOf(RFD)	I/$\overline{O}$ ↓ to RFD/$\overline{DAV}$ High Delay		500		500	
5	TdIOf(DAV)	I/$\overline{O}$ ↓ to RFD/$\overline{DAV}$ ↓ Delay	3		3		1
6	TdDO(IO)	I/$\overline{O}$ ↓ to Data Bus Driven	2		2		1

NOTES:
1. Units equal to $\overline{AS}$ cycles.
2. Minimum delay is four $\overline{AS}$ cycles or one $\overline{AS}$ cycle after the corresponding IP is cleared, whichever is longer.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8036	CE	4.0 MHz	Z-CIO (40-pin)	Z8036A	CE	6.0 MHz	Z-CIO (40-pin)
Z8036	CM	4.0 MHz	Same as above	Z8036A	CM	6.0 MHz	Same as above
Z8036	CMB	4.0 MHz	Same as above	Z8036A	CMB	6.0 MHz	Same as above
Z8036	CS	4.0 MHz	Same as above	Z8036A	CS	6.0 MHz	Same as above
Z8036	DE	4.0 MHz	Same as above	Z8036A	DE	6.0 MHz	Same as above
Z8036	DS	4.0 MHz	Same as above	Z8036A	DS	6.0 MHz	Same as above
Z8036	PE	4.0 MHz	Same as above	Z8036A	PE	6.0 MHz	Same as above
Z8036	PS	4.0 MHz	Same as above	Z8036A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, MB = -55°C to 125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8038 Z8000™ Z-FIO FIFO Input/ Output Interface Unit

Zilog

Product Specification

September 1983

Features

- 128-byte FIFO buffer provides asynchronous bidirectional CPU/CPU or CPU/peripheral interface, expandable to any width in byte increments by use of multiple FIOs.
- Interlocked 2-Wire or 3-Wire Handshake logic port mode; Z-BUS or non-Z-BUS interface.
- Pattern-recognition logic stops DMA transfers and/or interrupts CPU; preset byte count can initiate variable-length DMA transfers.
- Seven sources of vectored/nonvectored interrupt which include pattern-match, byte count, empty or full buffer status; a dedicated "mailbox" register with interrupt capability provides CPU/CPU communication.
- $\overline{\text{REQUEST}}/\overline{\text{WAIT}}$ lines control high-speed data transfers.
- All functions are software controlled via directly addressable read/write registers.

General Description

The Z8038 FIO provides an asynchronous 128-byte FIFO buffer between two CPUs or between a CPU and a peripheral device. This buffer interface expands to a 16-bit or wider data path and expands in depth to add as many Z8060 FIFOs (and an additional FIO) as are needed.

The FIO manages data transfers by assuming Z-BUS, non-Z-BUS microprocessor (a generalized microprocessor interface), Interlocked 2-Wire Handshake, and 3-Wire Handshake operating modes. These modes interface dissimilar CPUs or CPUs and peripherals running under differing speeds or protocols, allowing asynchronous data transactions and improving I/O overhead by as much as two orders of magnitude. Figures 1 and 2 show how the signals controlling these operating modes are mapped to the FIO pins.

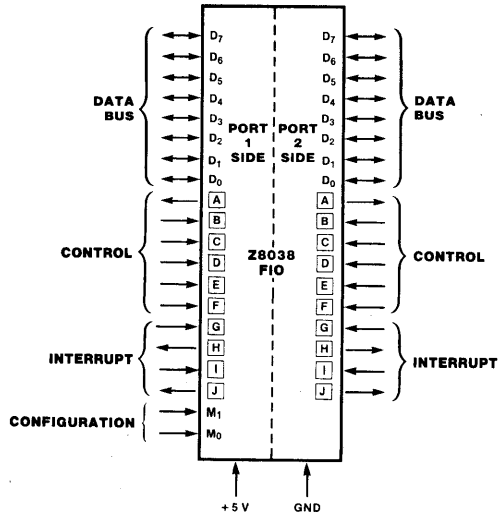


Figure 1. Pin Functions

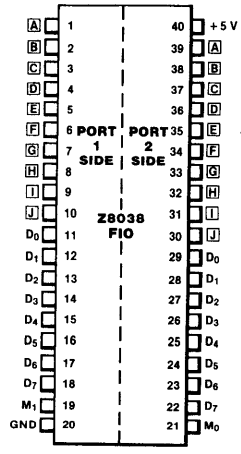


Figure 2. Pin Assignments

General Description (Continued)

The FIO supports the Z-BUS interrupt protocols, generating seven sources of interrupts upon any of the following events: a write to a message register, change in data direction, pattern match, status match, over/underflow error, buffer full and buffer empty status. Each interrupt source can be enabled or disabled, and can also place an interrupt vector on the port address/data lines.

The data transfer logic of the FIO has been specially designed to work with DMA (Direct Memory Access) devices for high-speed transfers. It provides for data transfers to or from memory each machine cycle, while the DMA device generates memory address and control signals. The FIO also supports the variably sized block length, improving system throughput when multiple variable length messages are transferred amongst several sources.

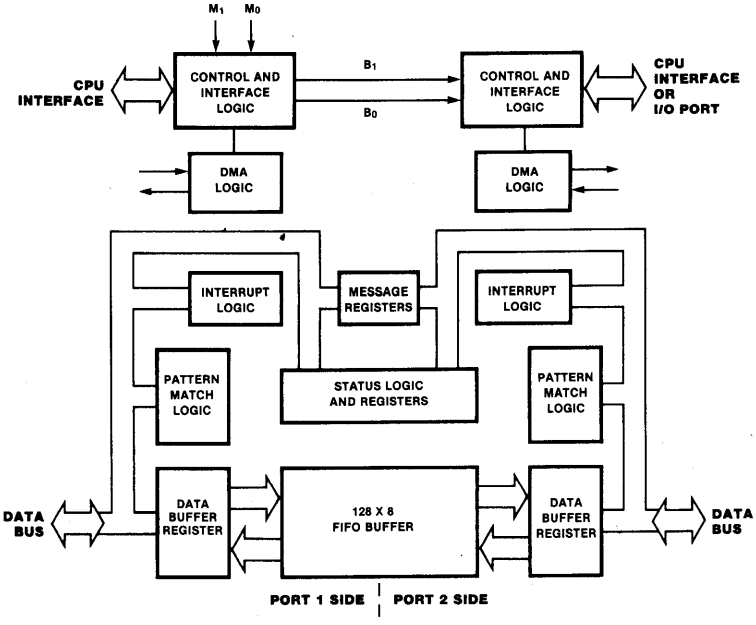


Figure 3. FIO Block Diagram

Functional Description

Operating Modes. Ports 1 and 2 operate in any of twelve combinations of operating modes, listed in Table 2. Port 1 functions in either the Z-BUS or non-Z-BUS microprocessor modes, while Port 2 functions in Z-BUS, non-Z-BUS, Interlocked 2-Wire Handshake, and 3-Wire Handshake modes. Table 1 describes the signals and their corresponding pins in each of these modes.

The pin diagrams of the FIO are identical, except for two pins on the Port 1 side, which select that port's operating mode. Port 2's operating mode is programmed by two bits in Port 1's Control register 0. Table 2 describes the combinations of operating modes; Table 3 describes the control signals mapped to pins A-J in the five possible operating modes.

Signal Pins	Z-BUS Low Byte	Z-BUS High Byte	Non-Z-BUS	Interlocked HS Port*	3-Wire HS Port*
A	$\overline{REQ}/\overline{WT}$	$\overline{REQ}/\overline{WT}$	$\overline{REQ}/\overline{WT}$	RFD/$\overline{DAV}$	RFD/$\overline{DAV}$
B	$\overline{DMASTB}$	$\overline{DMASTB}$	$\overline{DACK}$	$\overline{ACKIN}$	$\overline{DAV}$/DAC
C	$\overline{DS}$	$\overline{DS}$	$\overline{RD}$	FULL	DAC/RFD
D	R/$\overline{W}$	R/$\overline{W}$	$\overline{WR}$	EMPTY	EMPTY
E	$\overline{CS}$	$\overline{CS}$	$\overline{CE}$	$\overline{CLEAR}$	$\overline{CLEAR}$
F	$\overline{AS}$	AS	C/$\overline{D}$	DATA DIR	DATA DIR
G	$\overline{INTACK}$	A_0	$\overline{INTACK}$	IN_0	IN_0
H	IEO	A_1	IEO	OUT_1	OUT_1
I	IEI	A_2	IEI	$\overline{OE}$	$\overline{OE}$
J	$\overline{INT}$	A_3	$\overline{INT}$	OUT_3	OUT_3

*2 side only.

Table 1. Pin Assignments

Functional Description (Continued)

Mode	M_1	M_0	B_1	B_0	Port 1	Port 2
0	0	0	0	0	Z-BUS Low Byte	Z-BUS Low Byte
1	0	0	0	1	Z-BUS Low Byte	Non-Z-BUS
2	0	0	1	0	Z-BUS Low Byte	3-Wire Handshake
3	0	0	1	1	Z-BUS Low Byte	2-Wire Handshake
4	0	1	0	0	Z-BUS High Byte	Z-BUS High Byte
5	0	1	0	1	Z-BUS High Byte	Non-Z-BUS
6	0	1	1	0	Z-BUS High Byte	3-Wire Handshake
7	0	1	1	1	Z-BUS High Byte	2-Wire Handshake
8	1	0	0	0	Non-Z-BUS	Z-BUS Low Byte
9	1	0	0	1	Non-Z-BUS	Non-Z-BUS
10	1	0	1	0	Non-Z-BUS	3-Wire Handshake
11	1	0	1	1	Non-Z-BUS	2-Wire Handshake

Table 2. Operating Modes

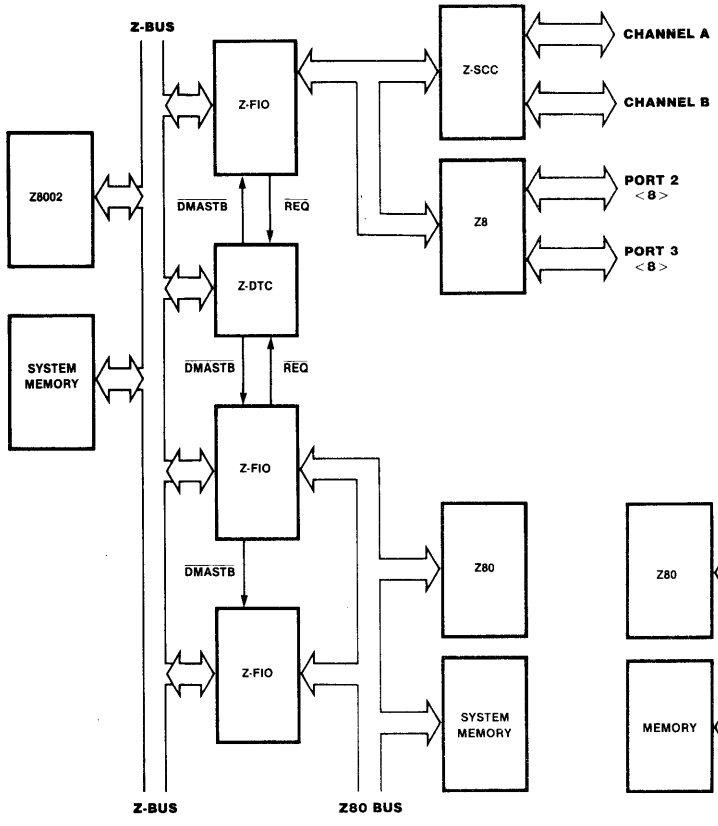


Figure 4. CPU to CPU Configuration

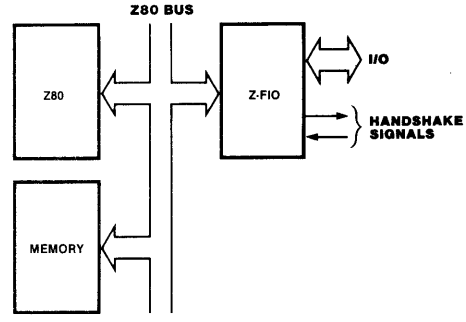


Figure 5. CPU to I/O Configuration

Pins Common To Both Sides

Pin Signals	Pin Names	Pin Numbers	Signal Description
M_0	M_0	21	M_1 and M_0 program Port 1 side CPU interface
M_1	M_1	19	
+5 Vdc	+5 Vdc	40	DC power source
GND	GND	20	DC power ground

Z-BUS Low Byte Mode

Pin Signals	Pin Names	Pin Numbers Port 1	Pin Numbers Port 2	Signal Description
AD_0-AD_7 (Address/Data)	D_0-D_7	11-18	29-22	Multiplexed bidirectional address/data lines, Z-BUS compatible.
$\overline{REQ}/\overline{WAIT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfers.
$\overline{DMASTB}$ (Direct Memory Access Strobe)	B	2	38	Input, active Low. Strobes DMA data to and from the FIFO buffer.
$\overline{DS}$ (Data Strobe)	C	3	37	Input, active Low. Provides timing for data transfer to or from FIO.
$R/\overline{W}$ (Read/Write)	D	4	36	Input; active High signals CPU read from FIO; active Low signals CPU write to FIO.
$\overline{CS}$ (Chip Select)	E	5	35	Input, active Low. Enables FIO. Latched on the rising edge of $\overline{AS}$.
$\overline{AS}$ (Address Strobe)	F	6	34	Input, active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ sampled while $\overline{AS}$ Low.
$\overline{INTACK}$ (Interrupt Acknowledge)	G	7	33	Input, active Low. Acknowledges an interrupt. Latched on the rising edge of $\overline{AS}$.
IEO (Interrupt Enable Out)	H	8	32	Output, active High. Sends interrupt enable to lower priority device IEI pin.
IEI (Interrupt Enable In)	I	9	31	Input, active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (Interrupt)	J	10	30	Output, open drain, active Low. Signals FIO interrupt request to CPU.

Z-BUS High Byte Mode

Pin Signals	Pin Names	Pin Numbers Port 1	Pin Numbers Port 2	Signal Description
AD_0-AD_7 (Address/Data)	D_0-D_7	11-18	29-22	Multiplexed bidirectional address/data lines, Z-BUS compatible.
$\overline{REQ}/\overline{WAIT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfers.
$\overline{DMASTB}$ (Direct Memory Access Strobe)	B	2	38	Input, active Low. Strobes DMA data to and from the FIFO buffer.
$\overline{DS}$ (Data Strobe)	C	3	37	Input, active Low. Provides timing for transfer of data to or from FIO.
$R/\overline{W}$ (Read/Write)	D	4	36	Input, active High. Signals CPU read from FIO; active Low signals CPU write to FIO.
$\overline{CS}$ (Chip Select)	E	5	35	Input, active Low. Enables FIO. Latched on the rising edge of $\overline{AS}$.
$\overline{AS}$ (Address Strobe)	F	6	34	Input, active Low. Addresses, $\overline{CS}$ and $\overline{INTACK}$ are sampled while $\overline{AS}$ is Low.
A_0 (Address Bit 0)	G	7	33	Input, active High. With A_1, A_2, and A_3, addresses FIO internal registers.
A_1 (Address Bit 1)	H	8	32	Input, active High. With A_0, A_2, and A_3, addresses FIO internal registers.
A_2 (Address Bit 2)	I	9	31	Input, active High. With A_0, A_1, and A_3, addresses FIO internal registers.
A_3 (Address Bit 3)	J	10	30	Input, active High. With A_0, A_1, and A_2, addresses FIO internal registers.

Table 3. Signal/Pin Descriptions

Non-Z-BUS Mode

Pin Signals	Pin Names	Pin Numbers Port 1	Port 4	Signal Description
D_0-D_7 (Data)	D_0-D_7	11-18	29-22	Bidirectional data bus.
$\overline{REQ}/\overline{WT}$ (Request/Wait)	A	1	39	Output, active Low, REQUEST (ready) line for DMA transfer; $\overline{WAIT}$ line (open-drain) output for synchronized CPU and FIO data transfer.
$\overline{DACK}$ (DMA Acknowledge)	B	2	38	Input, active Low. DMA acknowledge.
$\overline{RD}$ (Read)	C	3	37	Input, active Low. Signals CPU read from FIO.
$\overline{WR}$ (Write)	D	4	36	Input, active Low. Signals CPU write to FIO.
$\overline{CE}$ (Chip Select)	E	5	35	Input, active Low. Used to select FIO.
C/$\overline{D}$ (Control/Data)	F	6	34	Input, active High. Identifies control byte on D_0-D_7; active Low identifies data byte on D_0-D_7.
$\overline{INTACK}$ (Interrupt Acknowledge)	G	7	33	Input, active Low. Acknowledges an interrupt.
IEO (Interrupt Enable Out)	H	8	32	Output, active High. Sends interrupt enable to lower priority device IEI pin.
IEI (Interrupt Enable In)	I	9	31	Input, active High. Receives interrupt enable from higher priority device IEO signal.
$\overline{INT}$ (Interrupt)	J	10	30	Output, open drain, active Low. Signals FIO interrupt to CPU.

Port 2—I/O Port Mode

Pin Signals	Pin Names	Pin Numbers	Mode	Signal Description
D_0-D_7 (Data)	D_0-D_7	29-22	2-Wire HS* 3-Wire HS	Bidirectional data bus.
RFD/$\overline{DAV}$ (Ready for Data/Data Available)	A	39	2-Wire HS 3-Wire HS	Output, RFD active High. Signals peripherals that FIO is ready to receive data. $\overline{DAV}$ active Low signals that FIO is ready to send data to peripherals.
$\overline{ACKIN}$ (Acknowledge Input)	B	38	2-Wire HS	Input, active Low. Signals FIO that output data is received by peripherals or that input data is valid.
$\overline{DAV}$/DAC (Data Available/Data Accepted)	B	38	3-Wire HS	Input; $\overline{DAV}$ (active Low) signals that data is valid on bus. DAC (active High) signals that output data is accepted by peripherals.
FULL	C	37	2-Wire HS	Output, open drain, active High. Signals that FIO buffer is full.
DAC/RFD (Data Accepted/Ready for Data)	C	37	3-Wire HS	Direction controlled by internal programming. Both active High. DAC (an output) signals that FIO has received data from peripheral; RFD (an input) signals that the listeners are ready for data.
EMPTY	D	36	2-Wire HS 3-Wire HS	Output, open drain, active High. Signals that FIFO buffer is empty.
$\overline{CLEAR}$	E	35	2-Wire HS 3-Wire HS	Programmable input or output, active Low. Clears all data from FIFO buffer.
DATA DIR (Data Direction)	F	34	2-Wire HS 3-Wire HS	Programmable input or output. Active High signals data input to Port 2; Low signals data output from Port 2.
IN_0	G	33	2-Wire HS 3-Wire HS	Input line to D_0 of Control Register 3.
OUT_1	H	32	2-Wire HS 3-Wire HS	Output line from D_1 of Control Register 3.
$\overline{OE}$ (Output Enable)	I	31	2-Wire HS 3-Wire HS	Input, active Low. When Low, enables bus drivers. When High, floats bus drivers at high impedance.
OUT_3	J	30	2-Wire HS 3-Wire HS	Output line from D_3 of Control register 3.

*Handshake

Table 3. Signal/Pin Descriptions (Continued)

Reset

The FIO can be reset under either hardware or software control by one of the following methods:

- By forcing both $\overline{AS}$ and $\overline{DS}$ Low simultaneously in Z-BUS mode (normally illegal).
- By forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously in non-Z-BUS mode.
- By writing a 1 to the Reset bit in Control register 0 for software reset.

In the Reset state, all control bits are cleared to 0. Only after clearing the Reset bit (by writing a 0 to it) can the other command bits be programmed. This action is true for both sides of the FIO when programmed as a CPU interface.

For proper system control, when Port 1 is reset, Port 2 is also reset. In addition, all Port 2's outputs are floating and all inputs are ignored. To initiate the data transfer, Port 2 must be enabled by Port 1. The Port 2 CPU can determine when it is enabled by reading Control register 0, which reads "floating" data bus if not enabled and "01_H" if enabled.

CPU Interfaces

The FIO is designed to work with both Z-BUS- and non-Z-BUS-type CPUs on both Port 1 and Port 2. The Z-BUS configuration interfaces CPUs with time-multiplexed address and data information on the same pins. The Z8001, Z8002, and Z8 are examples of this type of CPU. The $\overline{AS}$ (Address Strobe) pin is used to latch the address and chip select information sent out by the CPU. The R/$\overline{W}$ (Read/Write) pin and the $\overline{DS}$ (Data Strobe) pin are used for timing reads and writes from the CPU to the FIO (Figures 6 and 7).

The non-Z-BUS configuration is used for CPUs where the address and data buses are separate. Examples of this type of CPU are the Z80 and 8080. The $\overline{RD}$ (Read) and $\overline{WR}$ (Write) pins are used to time reads and writes from the CPU to the FIO (Figures 9 and 10). The C/$\overline{D}$ (Control/Data) pin is used to directly access the FIFO buffer (C/$\overline{D}$ = 0) and to access the other registers (C/$\overline{D}$ = 1). Read and write to all

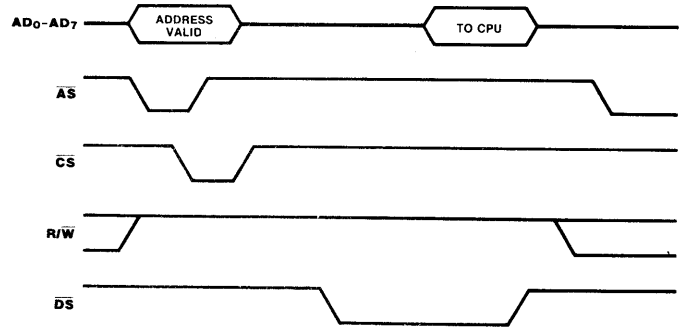


Figure 6. Z-BUS Read Cycle Timing

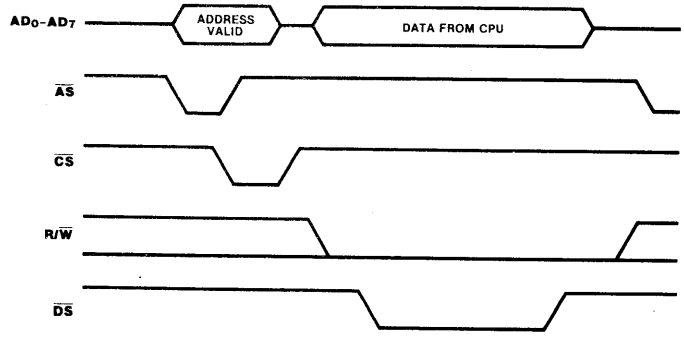


Figure 7. Z-BUS Write Cycle Timing

CPU Interfaces (Continued)

registers except the FIFO buffer[1] are two-step operations, described as follows (Figure 8). First, write the address ($C/\overline{D} = 1$) of the register to be accessed into the Pointer Register (State 0); second, read or write ($C/\overline{D} = 1$) to the register pointed at previously (State 1). Continuous status monitoring can be performed in State 1 by continuous Control Read operations ($C/\overline{D} = 1$).

[1]*The FIFO buffer can also be accessed by this two-step operation.*

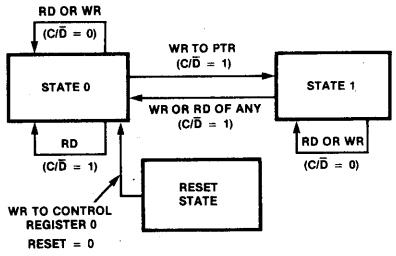


Figure 8. Register Access in Non-Z-BUS Mode

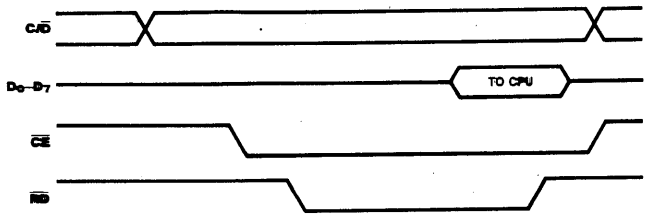


Figure 9. Non-Z-BUS Read Cycle Timing

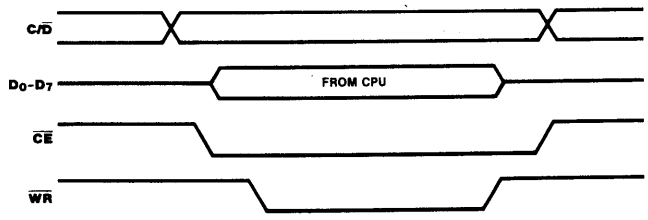


Figure 10. Non-Z-BUS Write Cycle Timing

$\overline{\text{WAIT}}$ Operation

When data is output by the CPU, the $\overline{\text{REQ}}/\overline{\text{WT}}$ ($\overline{\text{WAIT}}$) pin is active (Low) only when the FIFO buffer is full, the chip is selected, and the FIFO buffer is addressed. $\overline{\text{WAIT}}$ goes inactive when the FIFO buffer is not full.

When data is input by the CPU, the $\overline{\text{REQ}}/\overline{\text{WT}}$ pin becomes active (Low) only when the FIFO buffer is empty, the chip is selected, and the FIFO buffer is addressed. $\overline{\text{WAIT}}$ goes inactive when the FIFO buffer is not empty.

Interrupt Operation

The FIO supports Zilog's prioritized daisy chain interrupt protocol for both Z-BUS and non-Z-BUS operating modes (for more details refer to the Zilog *Z-BUS Summary*).

Each side of the FIO has seven sources of interrupt. The priorities of these devices are fixed in the following order (highest to lowest): Mailbox Message, Change in Data Direction, Pattern Match, Status Match, Overflow/Underflow Error, Buffer Full, and Buffer Empty. Each interrupt source has three bits that control how it generates the interrupt. These bits are Interrupt Pending (IP), Interrupt Enable (IE), and Interrupt Under Service (IUS).

In addition, each side of the FIO has an interrupt vector and four bits controlling the FIO interrupt logic. These bits are Vector

Interrupt Operation (Continued)

Includes Status (VIS), Master Interrupt Enable (MIE), Disable Lower Chain (DLC), and No Vector (NV).

A typical Interrupt Acknowledge cycle for Z-BUS operation is shown in Figure 11 and for non-Z-BUS operation in Figure 12. The only difference is that in Z-BUS mode, $\overline{INTACK}$ is latched by $\overline{AS}$, and in non-Z-BUS mode $\overline{INTACK}$ is not latched.

When MIE = 1, reading the vector always includes status, independent of the state of the VIS bit. In this way, when VIS = 0, all information can be obtained with one additional read, thus conserving vector space. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified.

In non-Z-BUS mode, the IPs do not get set while in State 1. Therefore, to minimize interrupt latency, the FIO should be left in State 0. In Z-BUS mode IPS are set by an $\overline{AS}$ following the event.

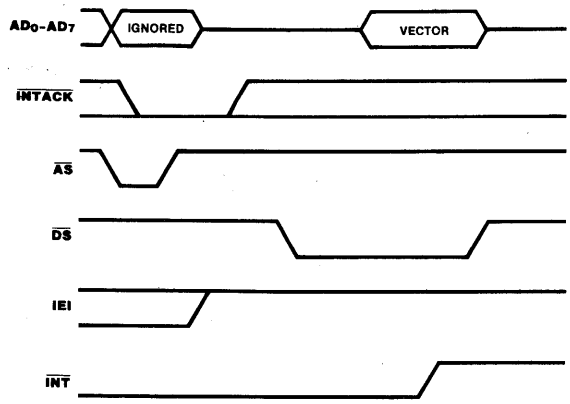


Figure 11. Z-BUS Interrupt Acknowledge Cycle

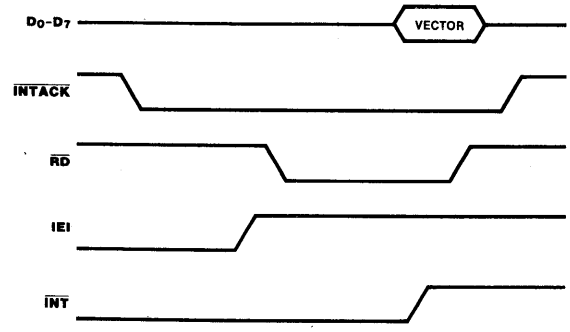


Figure 12. Non-Z-BUS Interrupt Acknowledge Cycle

CPU to CPU Operation

DMA Operation. The FIO is particularly well suited to work with a DMA in both Z-BUS and non-Z-BUS modes. A data transfer between the FIO and system memory can take place during every machine cycle on both sides of the FIO simultaneously.

In Z-BUS mode, the $\overline{DMASTB}$ pin (DMA Strobe) is used to read or write into the FIFO buffer. The R/$\overline{W}$ (Read/Write) and $\overline{DS}$ (Data Strobe) signals are ignored by the FIO; however, the $\overline{CS}$ (Chip Select) signal is not ignored and therefore must be kept invalid. Figures 13 and 14 show typical timing.

In Non-Z-BUS mode, the $\overline{DACK}$ pin (DMA Acknowledge) is used to tell the FIO that its DMA request is granted. After $\overline{DACK}$ goes Low, every read or write to the FIO goes into the FIFO buffer. Figures 15 and 16 show typical timing.

CPU to CPU Operation
(Continued)

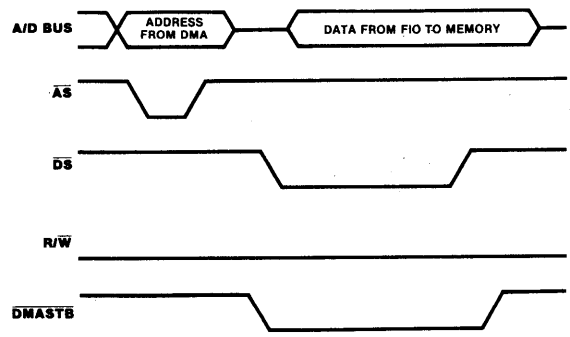


Figure 13. Z-BUS FIO to Memory Data Transaction

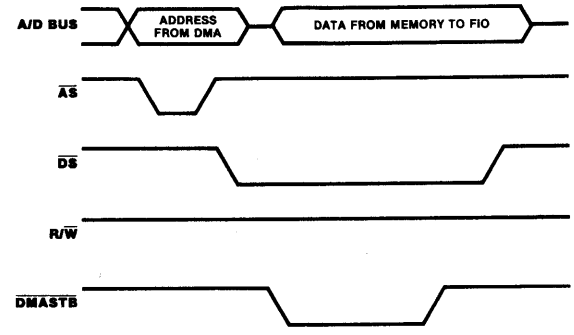


Figure 14. Z-BUS Memory to FIO Data Transaction

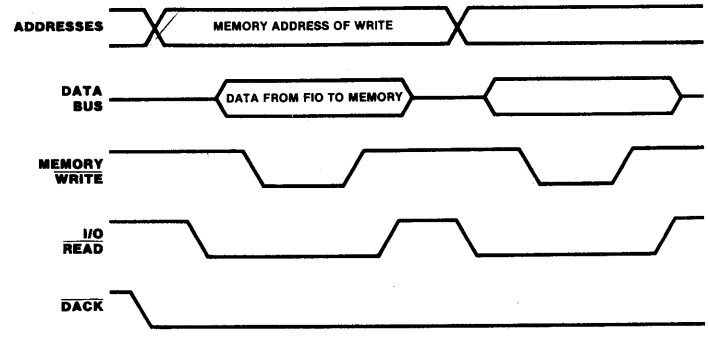


Figure 15. Non-Z-BUS FIO to Memory Transaction

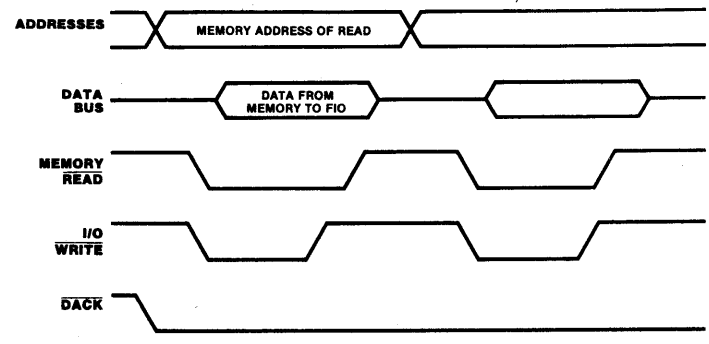
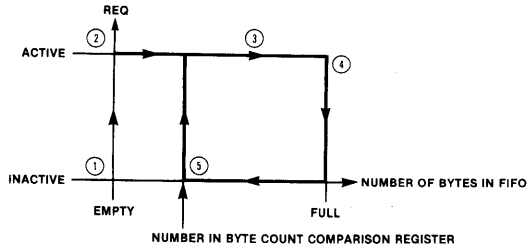


Figure 16. Non-Z-BUS Memory to FIO Data Transaction

CPU to CPU Operation (Continued)

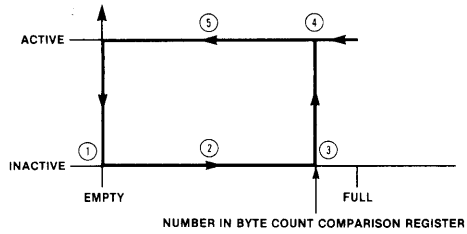
The FIO provides a special mode to enhance its DMA transfer capability. When data is written into the FIFO buffer, the $\overline{REQ}/\overline{WT}$ (REQUEST) pin is active (Low) until the FIFO buffer is full. It then goes inactive and stays inactive until the number of bytes in the FIFO buffer is equal to the value programmed into the Byte Count Comparison register. Then the REQUEST signal goes active and the sequence starts over again (Figure 17).



NOTES:
1. FIFO empty.
2. REQUEST enabled, FIO requests DMA transfer.
3. DMA transfers data into the FIO.
4. FIFO full, REQUEST inactive.
5. The FIFO empties from the opposite port until the number of bytes in the FIFO buffer is the same as the number programmed in the Byte Count Comparison register.

Figure 17. Byte Count Control: Write to FIO

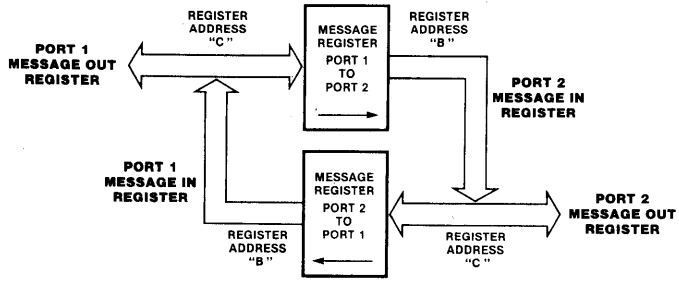
When data is read from the FIO, the $\overline{REQ}/\overline{WT}$ pin (REQUEST) is inactive until the number of bytes in the FIFO buffer is equal to the value programmed in the Byte Count Comparison register. The REQUEST signal then goes active and stays active until the FIFO buffer is empty. When empty, REQUEST goes inactive and the sequence starts over again (Figure 18).



NOTES:
1. FIFO empty.
2. CPU/DMA fills FIFO buffer from the opposite port.
3. Number of bytes in FIFO buffer is the same as the number of bytes programmed in the Byte Count Comparison register.
4. REQUEST goes active.
5. DMA transfers data out of FIFO until it is empty.

Figure 18. Byte Count Control: Read from FIO

Message Registers. Two CPUs can communicate through a dedicated "mailbox" register without involving the 128 × 8 bit FIFO buffer (Figure 19). This mailbox approach is useful for transferring control parameters between the interfacing devices on either side of the FIO without using the FIFO buffer. For example, when Port 1's CPU writes to the Message Out register, Port 2's message IP is set. If interrupts are enabled, Port 2's CPU is interrupted. Port 2's message IP status is readable from the Port 1 side. When Port 2's CPU reads the data from its Message In register, the Port 2 IP is cleared. Thus, Port 1's CPU can read when the message has been read and can now send another message or follow whatever protocol that is set up between the two CPU's. The same transfer can also be made from Port 2's CPU to Port 1's CPU.



NOTE: Usable only for CPU/CPU interface.

Figure 19. Message Register Operation

CPU to CPU Operation (Continued)

CLEAR (Empty) FIFO Operation. The $\overline{\text{CLEAR}}$ FIFO bit (active Low) clears the FIFO buffer of data. Writing a 0 to this bit empties the FIFO buffer, inactivates the REQUEST line, and disables the handshake (if programmed). The $\overline{\text{CLEAR}}$ bit does not affect any control or data register. To remove the $\overline{\text{CLEAR}}$ state, write a 1 to the $\overline{\text{CLEAR}}$ bit.

In CPU/CPU mode, under program control, only one of the ports can empty the FIFO by writing to its Control Register 3, bit 6. The Port 1 CPU must program bit 7 in Control Register 3 to determine which port controls the $\overline{\text{CLEAR}}$ FIFO operation (0 = Port 1 control; 1 = Port 2 control).

Direction of Data Transfer Operation. The Data Direction bit controls the direction of data transfer in the FIFO buffer. The Data Direction bit is defined as 0 = output from CPU and 1 = input to CPU. This bit reads correctly when read by either port's CPU. For example, if Port 1's CPU reads a 0 (CPU output) in its Data Direction bit, then Port 2's CPU reads a 1 (input to CPU) in its Data Direction bit.

In CPU/CPU mode, under program control, only one of the ports can control the direction of data transfer. The Port 1 CPU must program bit 5 in Control Register 3 to determine which port controls the data direction (0 = Port 1 control; 1 = Port 2 control). Figure 20 shows FIO data transfer options.

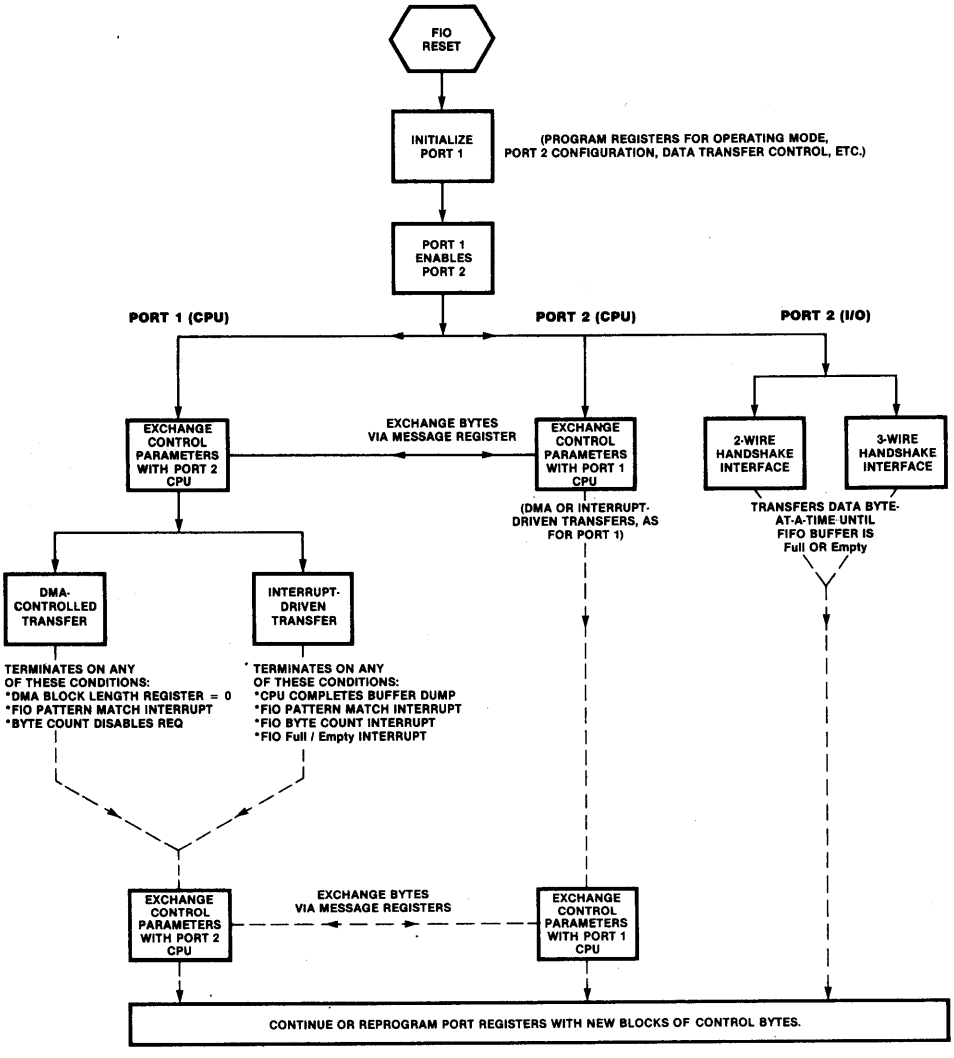


Figure 20. FIO Data Transfer Options

CPU to I/O Operation

When Port 2 is programmed in the Interlocked 2-Wire Handshake mode or the 3-Wire Handshake mode, and Port A is programmed in Z-BUS or non-Z-BUS Microprocessor mode, the FIO interfaces a CPU and a peripheral device. In the Interlocked 2-Wire Handshake mode, RFD/$\overline{\text{DAV}}$ and $\overline{\text{ACKIN}}$ strobe data to and from Port 2. In the 3-Wire Handshake mode, RFD/$\overline{\text{DAV}}$, $\overline{\text{DAV}}$/DAC, and DAC/RFD signals control data flow.

Interlocked 2-Wire Handshake. In the Interlocked Handshake, the action of the FIO must be acknowledged by the other half of the handshake before the next action can take place. In output mode, Port 2 does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, in input mode, Port 2 does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging Port 2's acceptance of the last byte. This allows the FIO to directly interface to a Z8's port, a CIO's port, a UPC's port, another FIO port, or another FIFO Z8060, with no external logic (Figures 21 and 22).

3-Wire Handshake. The 3-Wire Handshake is designed for applications in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate that an input port is ready for new data or that it has accepted the present data. In the 3-Wire Handshake, the rising edge of the RFD status line indicates that the port is ready for data, and the rising edge of the DAC status line indicates that the data has been accepted. With 3-Wire Handshake, the lines of many input ports can be bussed together with open-drain drivers and the output port knows when all of the ports are ready and have accepted the data. This handshake is the same handshake used in the IEEE-488 Instruments. Since the port's direction can be changed under software control, bidirectional IEEE-488-type transfers can be performed. Figures 23 and 24 show the timings associated with 3-Wire Handshake communications.

CLEAR FIFO Operation. In CPU-to-I/O operation, the CLEAR FIFO operation can be performed by the CPU side (Port 1) under software control as previously explained. The CLEAR FIFO operation can also be performed under hardware control by defining the $\overline{\text{CLEAR}}$ pin of Port 2 as an input (Control Register 3, bit 7 = 1).

For cascading purposes, the $\overline{\text{CLEAR}}$ pin can also be defined as an output (Control Register 3, bit 7 = 0), which reflects the current state of the $\overline{\text{CLEAR}}$ FIFO bit. It can then empty other FIOs or initialize other devices in the system.

Data Direction Control. In CPU-to-I/O mode, the direction of data transfer can be controlled by the CPU side (Port 1) under software control as previously explained. The data direction can also be determined by hardware control by defining the Data Direction pin of Port 2 as an input (Control Register 3, bit 5 = 1).

For cascading purposes, the Data Direction pin can also be defined as an output (Control Register 3, bit 5 = 0) pin which reflects the current state of the Data Direction bit. It can then be used to control the direction of data transfer for other FIOs or for external logic.

On the Port 2 side, when data direction is 0, Port 2 is in Output Handshake mode. When data direction is 1, Port 2 is in Input Handshake mode.

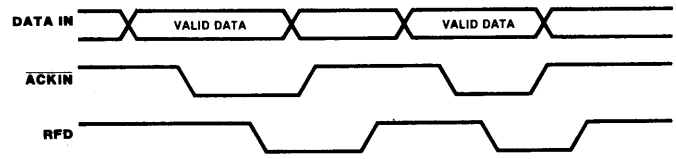


Figure 21. Interlocked Handshake Timing (Input) Port 2 Side Only

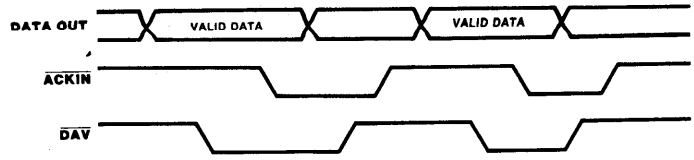


Figure 22. Interlocked Handshake Timing (Output) Port 2 Side Only

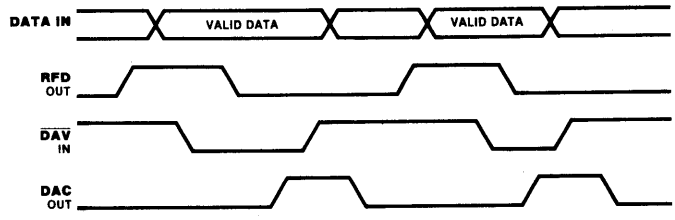


Figure 23. Input (Acceptor) Timing IEEE-488 HS Port: Port 2 Side Only

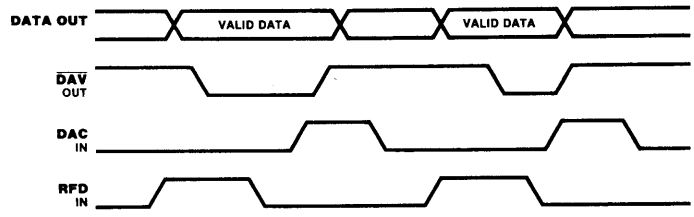


Figure 24. Output (Source) Timing IEEE-488 HS Port: Port 2 Side Only

Programming

The programming of the FIO is greatly simplified by the efficient grouping of the various operation modes in the control registers. Since all of the control registers are read/write, the need for maintaining their image in system memory is eliminated. Also, the read/write feature of the registers aids in system debugging.

Each side of the FIO has 16 registers. All 16 registers are used by the Port 1 side; Control register 2 is not used on the Port 2 side. All registers are addressable 0_H through F_H.

In the Z-BUS Low Byte mode, the FIO allows two methods for register addressing under control of the Right Justify Address (RJA) bit in Control register 0. When RJA = 0, address bus bits 1–4 are used for register addressing and bits 1, 5, 6, and 7 are ignored (Table 4). When RJA = 1, bits 0–3 are used for the register addresses, and bits 4–7 are ignored.

Control Registers. These four registers specify FIO operation. The Port 2 side control registers operate only if the Port 2 device is a CPU. The Port 2 CPU can control interface operations, including data direction, only when enabled by the setting of bit 0 in the Port 1 side of Control Register 2. A 1 in bit 1 of the same register enables the handshake logic.

Interrupt Status Registers. These four registers control and monitor the priority interrupt functions for the FIO.

Interrupt Vector Register. This register stores the interrupt service routine address. This vector is placed on D_0–D_7 when IUS is set by the Interrupt Acknowledge signal from the CPU. When bit 4 (Vector Includes Status) is set in Control Register 0, the reason for the interrupt is encoded within the vector address in bits 1, 2, and 3. If bit 5 is set in Control register 0, no vector is output by the FIO during an Interrupt Acknowledge cycle. However, IUS is set as usual.

Non Z-BUS	D_7-D_4	D_3	D_2	D_1	D_0	
Z-BUS High		A_3	A_2	A_1	A_0	
Z-BUS Low RJA=0	AD_7-AD_5	AD_4	AD_3	AD_2	AD_1	AD_0
Z-BUS Low RJA=1	AD_7-AD_4	AD_3	AD_2	AD_1	AD_0	
Description						
Control Register 0	x	0	0	0	0	x
Control Register 1	x	0	0	0	1	x
Interrupt Status Register 0	x	0	0	1	0	x
Interrupt Status Register 1	x	0	0	1	1	x
Interrupt Status Register 2	x	0	1	0	0	x
Interrupt Status Register 3	x	0	1	0	1	x
Interrupt Vector Register	x	0	1	1	0	x
Byte Count Register	x	0	1	1	1	x
Byte Count Comparison Register	x	1	0	0	0	x
Control Register 2*	x	1	0	0	1	x
Control Register 3	x	1	0	1	0	x
Message Out Register	x	1	0	1	1	x
Message In Register	x	1	1	0	0	x
Pattern Match Register	x	1	1	0	1	x
Pattern Mask Register	x	1	1	1	0	x
Data Buffer Register	x	1	1	1	1	x

x = Don't Care
*Register is only on Port 1 side

Table 4. FIO Register Address Summary

Programming (Continued)

Byte Count Compare Register. This register contains a value compared with the byte count in the Byte Count register. If the Byte Count Compare interrupt is enabled, an interrupt will occur upon compare.

Message Out Register. Either CPU can place a message in its Message Out register. If the opposite side Message register interrupt is enabled, the receiving side CPU will receive an interrupt request, advising that a message is present in its Message In register. Bit 5 in Control Register 1 on the initiating side is set when a message is written. It is cleared when the message is read by the receiving CPU.

Message In Register. This register receives a message placed in the Message Out register by the opposite side CPU.

Pattern Match Register. This register contains a bit pattern matched against the byte in the Data Buffer register. When these patterns match, a Pattern Match interrupt will be generated, if previously enabled.

Pattern Mask Register. The Pattern Mask register may be programmed with a bit pattern mask that limits comparable bits in the Pattern Match register to non-masked bits (1 = mask).

Data Buffer Register. This register contains the data to be read from or written to the FIFO buffer.

Byte Count Register. This is a read-only register, containing the byte count for the FIFO buffer. The byte count is derived by subtracting the number of bytes read from the buffer from the number of bytes written into the buffer. The count is "frozen" for an accurate reading by setting bit 6 (Freeze Status register) in Control Register 1. This bit is cleared when the Byte Count register read is completed.

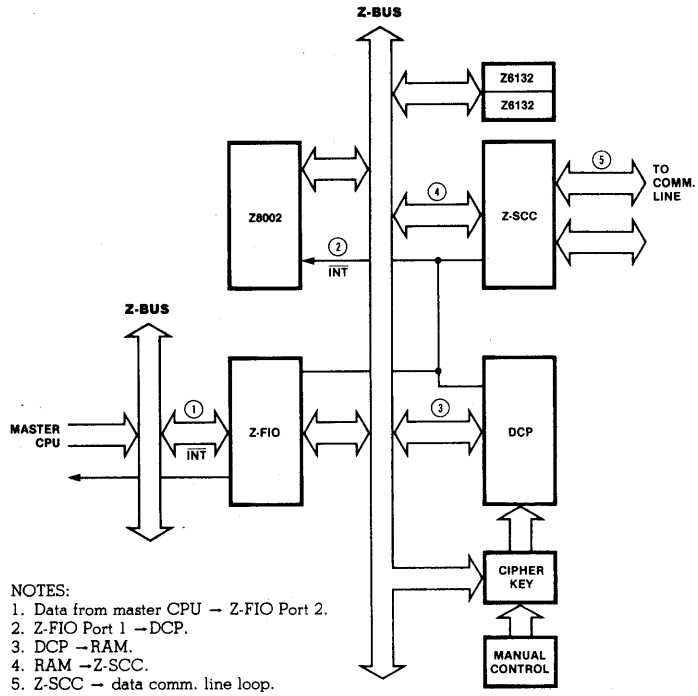
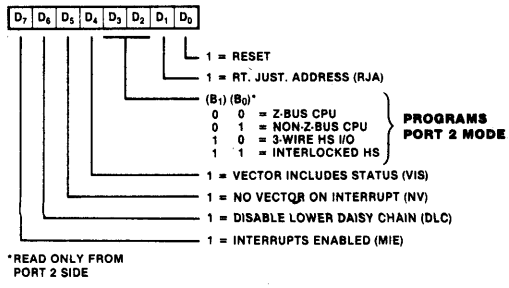


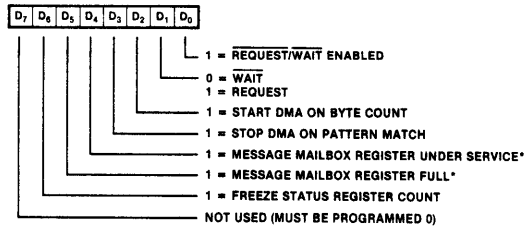
Figure 25. Typical Application: Node Controller

Control Register 0
Address: 0000
(Read/Write)



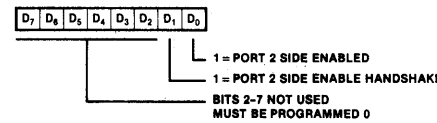
*READ ONLY FROM PORT 2 SIDE

Control Register 1
Address: 0001
(Read/Write)



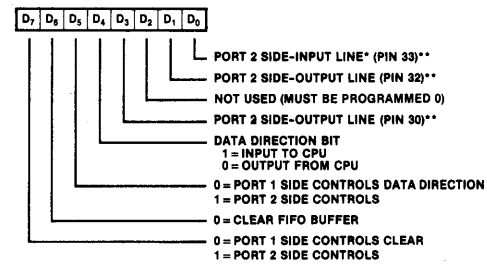
*READ-ONLY BITS

Control Register 2*
Address: 1001
(Read/Write)



*THIS REGISTER READS ALL 0'S FROM PORT 2 SIDE

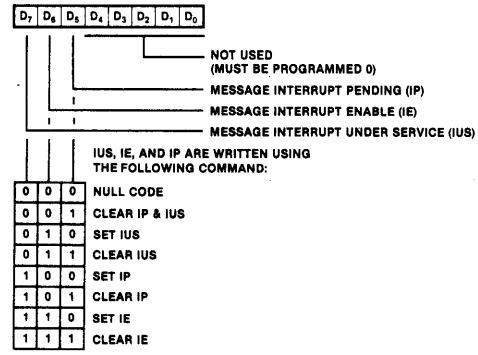
Control Register 3
Address: 1010
(Read/Write)



*READ-ONLY BITS
**ONLY WHEN PORT 2 IS AN I/O PORT

Figure 26. Control Registers

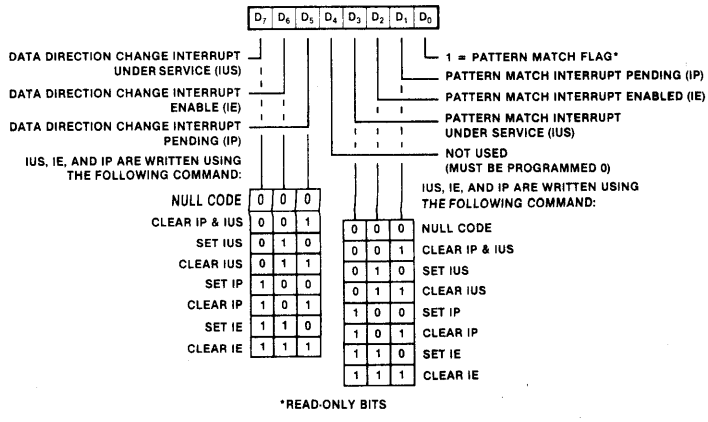
Interrupt Status Register 0
Address: 0010
(Read/Write)



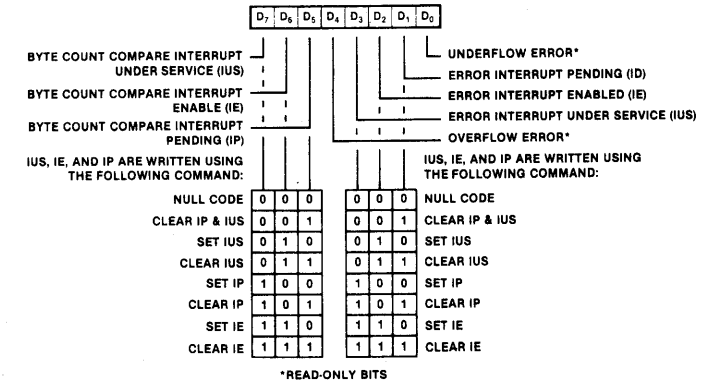
0	0	0	NULL CODE
0	0	1	CLEAR IP & IUS
0	1	0	SET IUS
0	1	1	CLEAR IUS
1	0	0	SET IP
1	0	1	CLEAR IP
1	1	0	SET IE
1	1	1	CLEAR IE

Figure 27. Interrupt Status Registers

Interrupt Status Register 1
Address: 0011
(Read/Write)



Interrupt Status Register 2
Address: 0100
(Read/Write)



Interrupt Status Register 3
Address: 0101
(Read/Write)

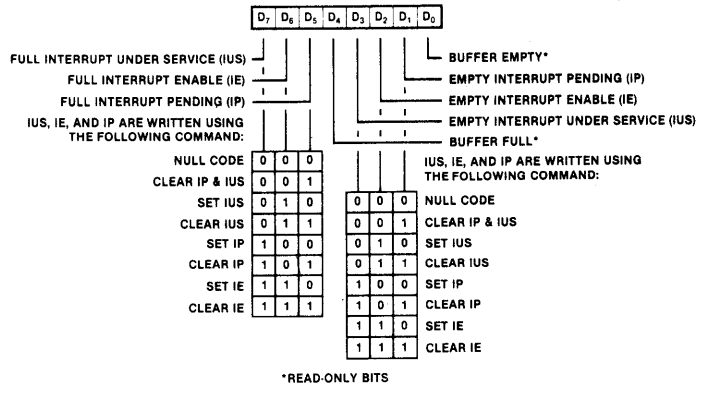


Figure 27. Interrupt Status Registers (Continued)

Registers (Continued)

Byte Count Register
Address: 0111
(Read Only)

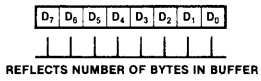


Figure 28. Byte Count Register

Interrupt Vector Register
Address: 0110
(Read/Write)

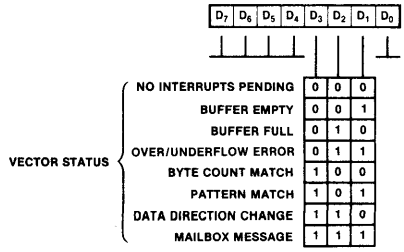


Figure 29. Interrupt Vector Register

Pattern Match Register
Address: 1101
(Read/Write)

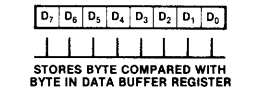


Figure 30. Pattern Match Register

Pattern Mask Register
Address: 1110
(Read/Write)

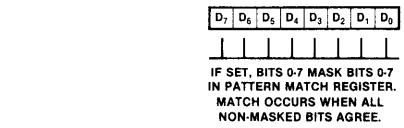


Figure 31. Pattern Mask Register

Data Buffer Register
Address: 1111
(Read/Write)

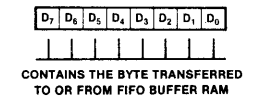


Figure 32. Data Buffer Register

Byte Count Comparison Register
Address: 1000
(Read/Write)

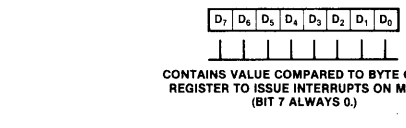


Figure 33. Byte Count Comparison Register

Message Out Register
Address: 1011
(Read/Write)

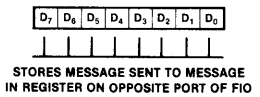


Figure 34. Message Out Register

Message In Register
Address: 1100
(Read Only)

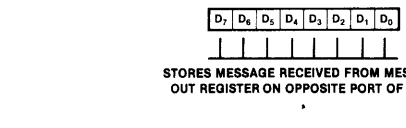


Figure 35. Message In Register

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . 0°C to +70°C

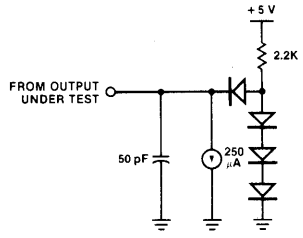
Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

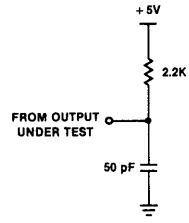
Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- T_A as specified in Ordering Information



Standard Test Load



Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
			0.5	V	I_{OL} = +3.2 mA
I_{IL}	Input Leakage	−10.0	+10.0	µA	0.4 ≤ V_{IN} ≤ +2.4V
I_{OL}	Output Leakage	−10.0	+10.0	µA	0.4 ≤ V_{OUT} ≤ +2.4V
I_{LM}	Mode Pins Input Leakage (Pins 19 and 21)	−100	+10.0	µA	0 < V_{IN} < V_{CC}
I_{CC}	V_{CC} Supply Current		200	mA	

V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

Inputs

tr	Any Input Rise Time	100	ns	
tf	Any Input Fall Time	100	ns	

f = 1 MHz, over specified temperature range.

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwAS	$\overline{AS}$ Low Width	70		50		1
2	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
3	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
4	TsCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
5	ThCSO(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
6	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↑ Delay	60		40		1
7	TsA(DS)	Address to $\overline{DS}$ ↓ (with $\overline{AS}$ ↑ to $\overline{DS}$ ↓ = 60 ns)	120		100		
8	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
9	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
10	TwDS	$\overline{DS}$ Low Width	390		250		
11	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
12	TdDS(DRV)	$\overline{DS}$ (Read) ↓ to Address Data Bus Driven	0		0		
13	TdDSf(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay		250		180	
14	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		
15	TdDSr(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0		0		
16	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	2
17	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	55		40		
18	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
19	Trc	Valid Access Recovery Time	1000		650		3

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V from steady state with minimum as load and load and maximum dc load.
3. This is the delay from $\overline{DS}$ of one CIO access to $\overline{DS}$ of another FIO access (either read or write).

* All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0". All timings are preliminary and subject to change.

† Units in nanoseconds (ns).

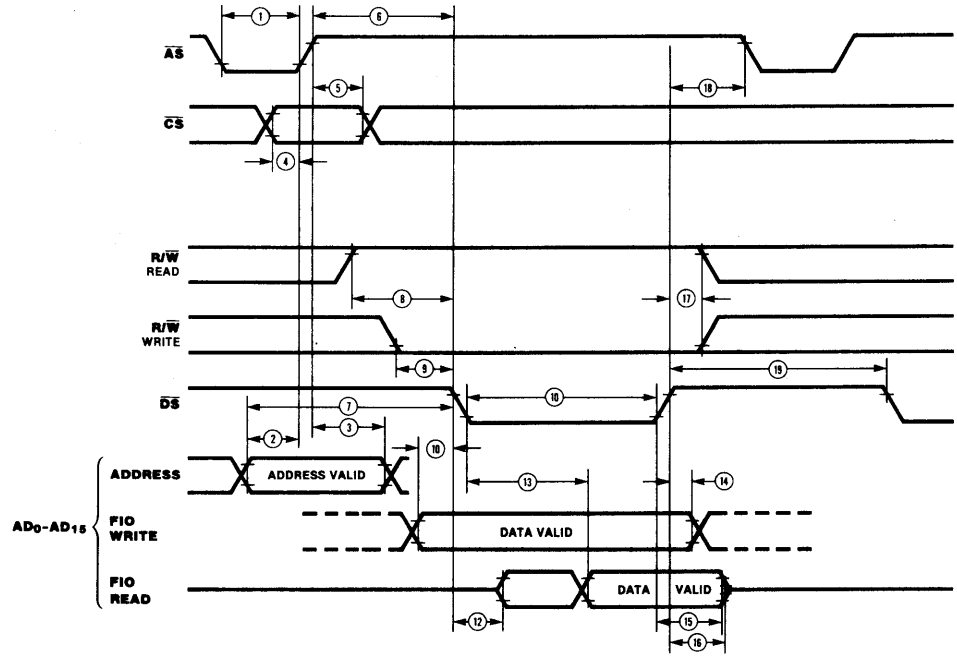


Figure 36. Z-BUS CPU Interface Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
20	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0			0	
21	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
22	TsDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Data Valid Delay		250		180	
23	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	390		250		
24	TdAS(IEO)	$\overline{AS}$ ↑ to IEO ↓ Delay ($\overline{INTACK}$ Cycle)		350		250	4
25	*TdIEI(IEO)*	*IEI to IEO Delay*		150		100	4
26	TsIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Setup Time	100		70		
27	ThIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↑ Hold Time	50		30		4
28	TdDS(INT)	$\overline{DS}$ ($\overline{INTACK}$ Cycle) to $\overline{INT}$ Delay		900		800	
29	TdDCST	Interrupt Daisy Chain Settle Time					4

NOTES:

4. The parameters for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{AS}$ to $\overline{DS}$ must be greater than the sum of TdAS(IEO) for the highest priority peripheral, TsIEI(DSA) for the lowest priority peripheral and TdIEI(IEO) for each peripheral, separating them in the chain.

* Timings are preliminary and subject to change.

† Units in nanoseconds (ns).

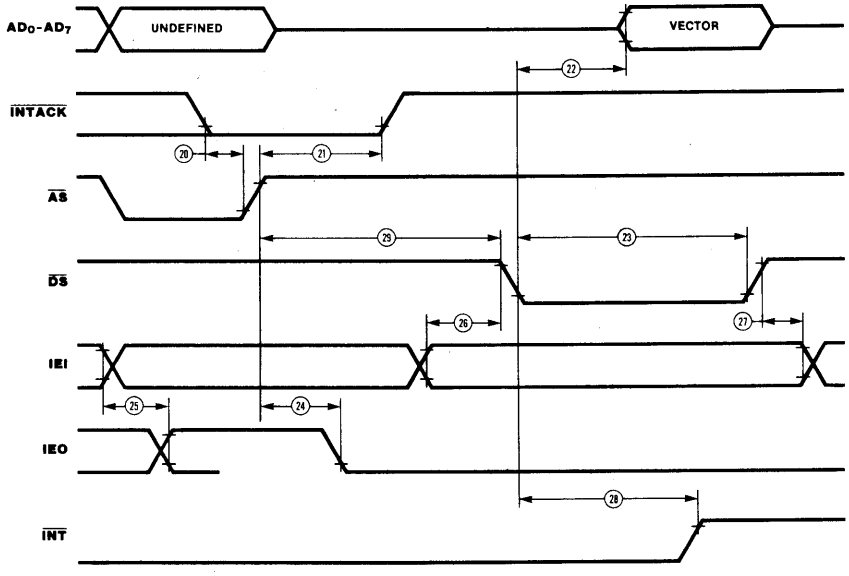


Figure 37. Z-BUS CPU Interrupt Acknowledge Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
30	TdMW(INT)	Message Write to $\overline{INT}$ Delay		1		1	5
31	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay		1		1	6
32	TdPMW(INT)	Pattern Match to $\overline{INT}$ Delay (Write Case)		1		1	
33	TdPMR(INT)	Pattern Match (Read Case) to $\overline{INT}$ Delay		1		1	
34	TdSC(INT)	Status Compare to $\overline{INT}$ Delay		1		1	6
35	TdER(INT)	Error to $\overline{INT}$ Delay		1		1	
36	TdEM(INT)	Empty to $\overline{INT}$ Delay		1		1	6
37	TdFL(INT)	Full to $\overline{INT}$ Delay		1		1	6
38	TdAS(INT)	$\overline{AS}$ to $\overline{INT}$ Delay					

NOTES:
5. Write is from the other side of FIO.
6. Write can be from either side, depending on programming of FIO.

* Timings are preliminary and subject to change.
† Units equal to $\overline{AS}$ Cycles + ns.

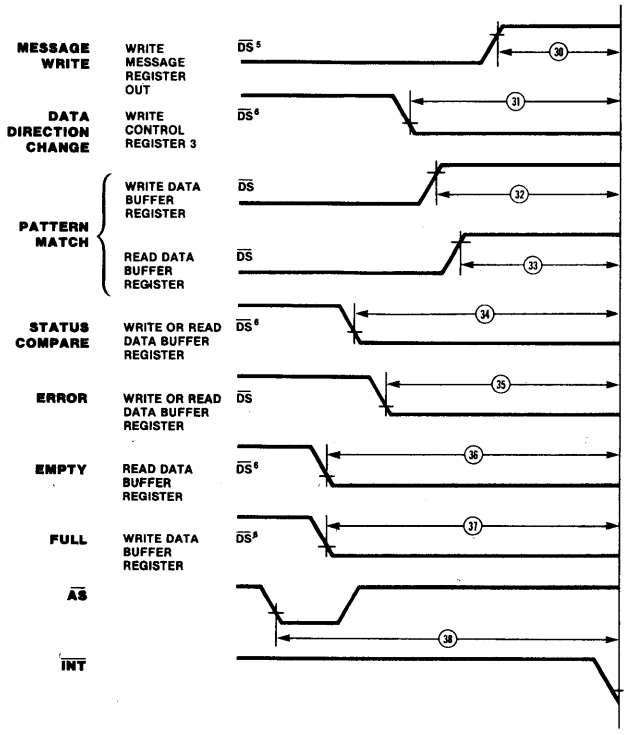


Figure 38. Z-BUS Interrupt Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDS(WAIT)	$\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Delay		190		160	
2	TdDSI(WAIT)	$\overline{DSI}$ ↑ to $\overline{WAIT}$ ↑ Delay		1000		1000	
3	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay		1000		1000	1
4	TdDS(REQ)	$\overline{DS}$ ↓ to $\overline{REQ}$ ↑ Delay		350		300	
5	TdDMA(REQ)	$\overline{DMASTB}$ ↓ to $\overline{REQ}$ ↑ Delay		350		300	
6	TdDSI(REQ)	$\overline{DSI}$ ↑ to $\overline{REQ}$ ↓ Delay		1000		1000	
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ ↓ Delay		1000		1000	
8	TdSU(DMA)	Data Setup Time to $\overline{DMASTB}$	200		150		
9	TdH(DMA)	Data Hold Time to $\overline{DMASTB}$	30		20		
10	TdDMA(DR)	$\overline{DMASTB}$ ↓ to Valid Data		150		100	
11	TdDMA(DRH)	$\overline{DMASTB}$ ↑ to Data Not Valid	0		0		
12	TdDMA(DR2)	$\overline{DMASTB}$ ↑ to Data Bus Float		70		45	

NOTES:
1. The delay is from $\overline{DAV}$ for 3-Wire Input Handshake. The delay is from DAC for 3-Wire Handshake.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

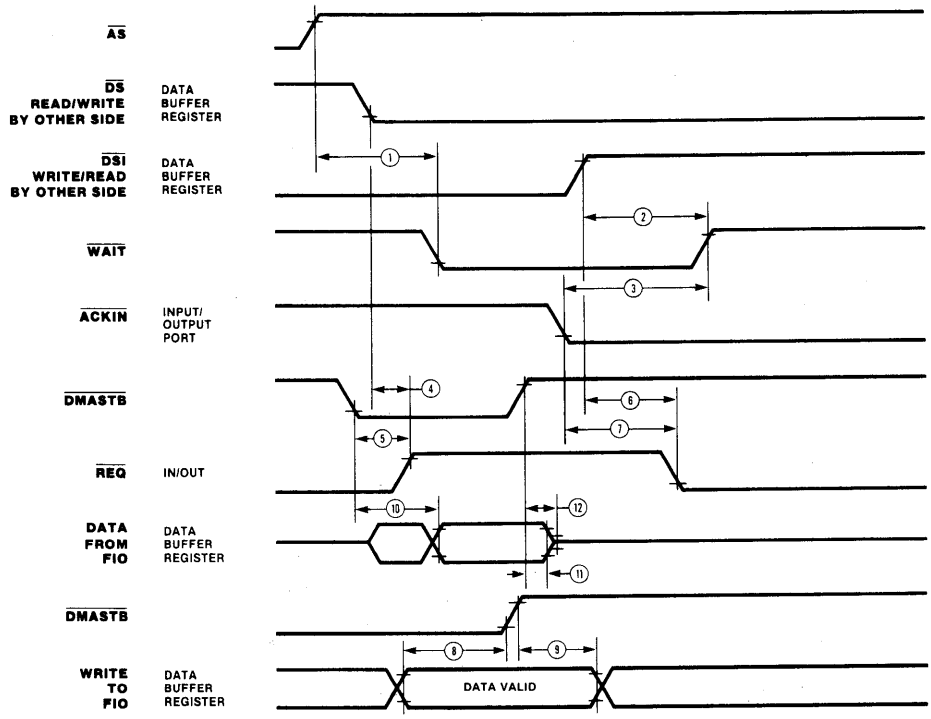


Figure 39. Z-BUS Request/Wait Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdDSQ(AS)	Delay from $\overline{DS}$ ↑ to $\overline{AS}$ ↓ for No Reset	40		20		
2	TdASQ(DS)	Delay for $\overline{AS}$ ↑ to $\overline{DS}$ ↓ for No Reset	50		30		
3	Tw(AS + DS)	Minimum Width of $\overline{AS}$ and $\overline{DS}$ Both Low for Reset.	500		350		1

NOTES:
1. Internal circuitry allows for the reset provided by the Z8 (DS held Low while $\overline{AS}$ pulses) to be sufficient.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

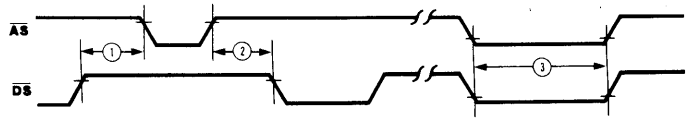


Figure 40. Z-BUS Reset Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsA(RD)	Address Setup to $\overline{RD}$ ↓	80		80		1
2	TsA(WR)	Address Setup to $\overline{WR}$ ↓	80		80		
3	ThA(RD)	Address Hold Time to $\overline{RD}$ ↑	0		0		1
4	ThA(WR)	Address Hold Time to $\overline{WR}$ ↑	0		0		
5	TsCEI(RD)	$\overline{CE}$ Low Setup Time to $\overline{RD}$	0		0		1
6	TsCEI(WR)	$\overline{CE}$ Low Setup Time to $\overline{WR}$	0		0		
7	ThCEI(RD)	$\overline{CE}$ Low Hold Time to $\overline{RD}$	0		0		1
8	ThCEI(WR)	$\overline{CE}$ Low Hold Time to $\overline{WR}$	0		0		
9	TsCEh(RD)	$\overline{CE}$ High Setup Time to $\overline{RD}$	100		70		1
10	TsCEh(WR)	$\overline{CE}$ High Setup Time to $\overline{WR}$	100		70		
11	TwRDl	$\overline{RD}$ Low Width	390		250		
12	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
13	TdRDf(DR)	$\overline{RD}$ ↓ to Valid Data Delay		250		180	
14	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
15	TdRD(DRz)	$\overline{RD}$ ↑ to Data Bus Float		70		45	2
16	TwWRl	$\overline{WR}$ Low Width	390		250		
17	TsDW(WR)	Data Setup Time to $\overline{WR}$	0		0		
18	ThDW(WR)	Data Hold Time to $\overline{WR}$	30		20		
19	Trc	Valid Access Recovery Time	1000		650		3

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time the output has changed 0.5 V from steady state with minimum ac load and maximum dc load.
3. This is the delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↑ of one FIO access to $\overline{RD}$ ↓ or $\overline{WR}$ ↓ of another FIO access.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

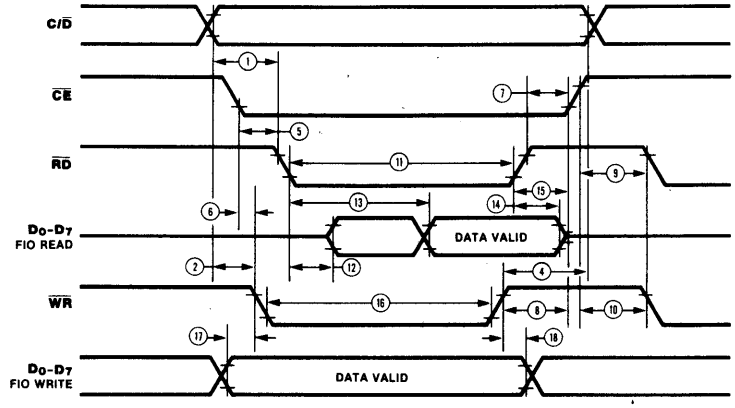


Figure 41. Non-Z-BUS CPU Interface Timing

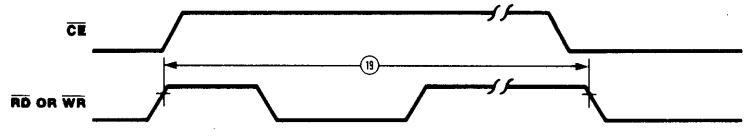


Figure 42. Non-Z-BUS Interface Timimg

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes†*
20	TdIEI(IEO)	IEI to IEO Delay		150		100	4
21	TdI(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		350		250	4
22	TsIEI(RDA)	IEI Setup Time to $\overline{RD}$ (Acknowledge)	100		70		4
23	TdRD(DR)	$\overline{RD}$ ↓ to Vector Valid Delay		250		180	
24	TwRDl(IA)	Read Low Width (Interrupt Acknowledge)	390		250		
25	ThIA(RD)	$\overline{INTACK}$ ↑ to $\overline{RD}$ ↑ Hold Time	30		20		
26	ThIEI(RD)	IEI Hold Time to $\overline{RD}$ ↑	20		10		
27	TdRD(INT)	$\overline{RD}$ ↑ to $\overline{INT}$ ↑ Delay		900		800	
28	TdDCST	Interrupt Daisy Chain Settle Time	350		250		4

NOTES:
4. The parameter for the devices in any particular daisy chain must meet the following constraint: The delay from $\overline{INTACK}$ ↓ to $\overline{RD}$ ↓ must be greater than the sum of TdINA(IEO) for the highest priority peripheral, TsIEI(RD) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.

† Units in nanoseconds (ns).

* Timings are preliminary and subject to change.

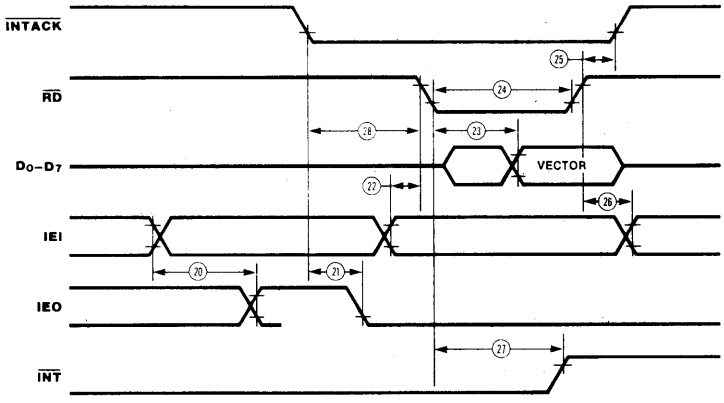


Figure 43. Non-Z-BUS Interrupt Acknowledge Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
29	TdMW(INT)	Message Write to $\overline{INT}$ Delay					5,6
30	TdDC(INT)	Data Direction Change to $\overline{INT}$ Delay					5,7
31	TdPMW(INT)	Pattern Match (Write Case) to $\overline{INT}$ Delay					5
32	TdPMR(INT)	Pattern Match (Read Case) to $\overline{INT}$ Delay					5
33	TdSC(INT)	Status Compare to $\overline{INT}$ Delay					5,7
34	TdER(INT)	Error to $\overline{INT}$ Delay					5,7
35	TdEM(INT)	Empty to $\overline{INT}$ Delay					5,7
36	TdFL(INT)	Full to $\overline{INT}$ Delay					5,7
37	TdSO(INT)	State 0 to $\overline{INT}$ Delay					

NOTES:
5. Delay number is valid for State 0 only.
6. Write is from other side of FIO.
7. Write can be from either side, depending on programming of FIO.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

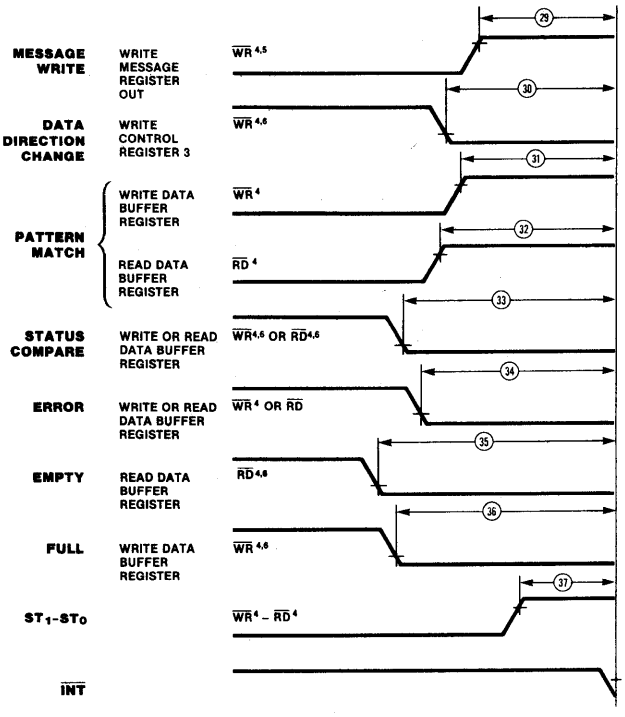


Figure 44. Z-FIO Non-Z-BUS Interrupt Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRD(WT)	$\overline{CE}$ ↓ to $\overline{WAIT}$ Active		200		170	
2	TdRD1(WT)	$\overline{RD}$1 ↑ or $\overline{WR}$1 ↑ to $\overline{WAIT}$ Inactive		1000		1000	
3	TdACK(WT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ Inactive		1000		1000	1
4	TdRD(REQ)	$\overline{RD}$ ↓ or $\overline{WR}$ ↓ to $\overline{REQ}$ Inactive		350		300	
5	TdRD1(REQ)	$\overline{RD}$1 ↑ or $\overline{WR}$1 ↑ to $\overline{REQ}$ Active		1000		1000	
6	TdACK(REQ)	$\overline{ACKIN}$ ↓ to $\overline{REQ}$ Active		1000		1000	
7	TdDAC(RD)	$\overline{DACK}$ ↓ to $\overline{RD}$ ↓ or $\overline{WR}$ ↓	100		80		
8	TSU(WR)	Data Setup Time to $\overline{WR}$	200				
9	Th(WR)	Data Hold Time to $\overline{WR}$	30			20	
10	TdDMA	$\overline{RD}$ ↓ to Valid Data		150		100	2
11	TdDMA(DRH)	$\overline{RD}$ ↑ to Data Not Valid	0		0		2
12	TdDMA(DRZ)	$\overline{RD}$ ↑ to Data Bus Float		70		45	2

NOTES:
1. The delay is from $\overline{DAV}$ ↓ for 3-Wire Input Handshake. The delay is from $\overline{DAC}$ ↑ for 3-Wire Input Handshake.
2. Only when $\overline{DACK}$ is active.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

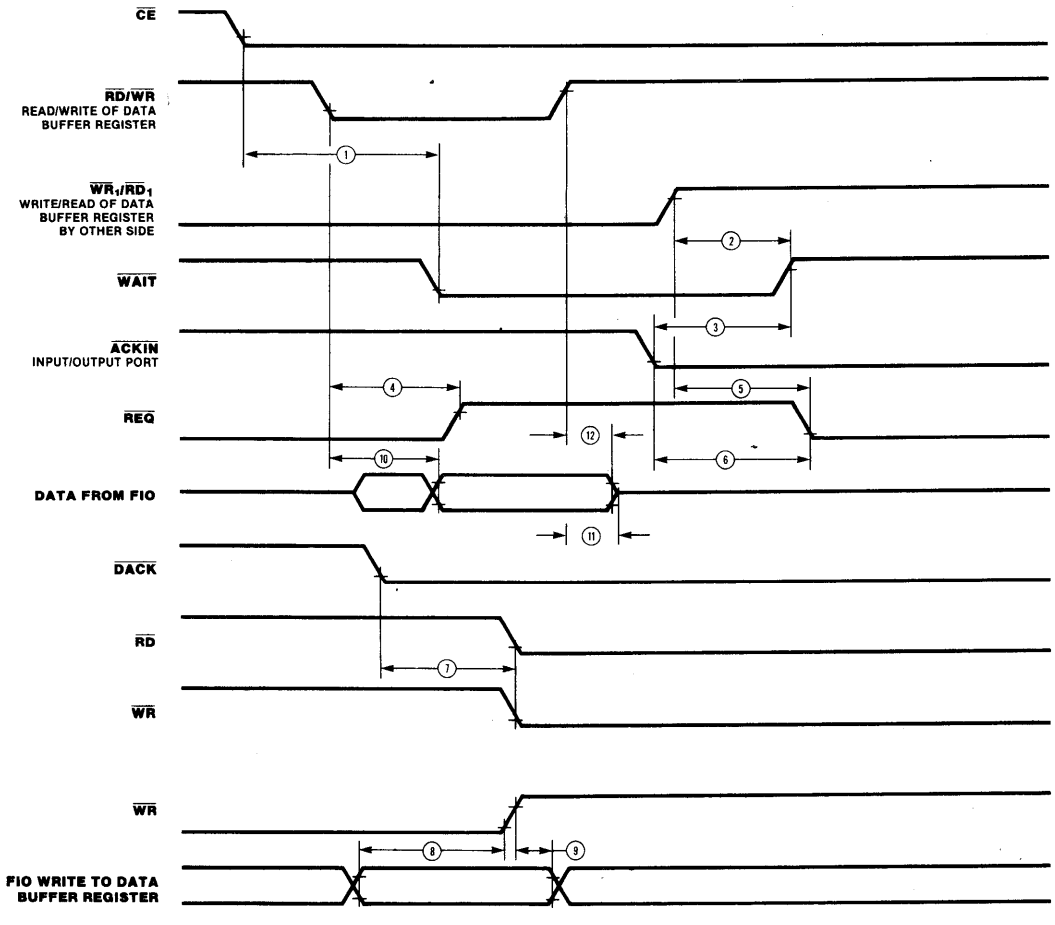


Figure 45. Non-Z-BUS Request/Wait Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓	100		70		
2	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓	100		70		
3	TwRD + WR	Width of $\overline{RD}$ and $\overline{WR}$, both Low for Reset	500		350		

NOTES:
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

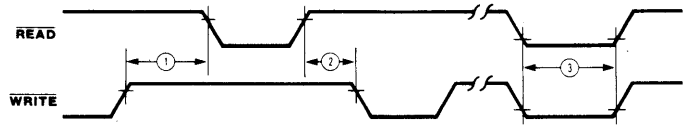


Figure 46. Non-Z-BUS Reset Timing

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwCLR	Width of Clear to Reset FIFO	700		700		
2	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven	0		0		
3	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float					

NOTES:
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

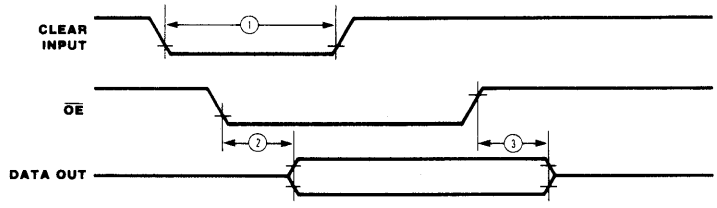


Figure 47. Port 2 Side Operation

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ to Setup Time	50		50		
2	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0	500	0	500	
3	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		0		
4	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	50		25		
5	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		0		
6	ThDO(ACK)	Data Out to $\overline{ACKIN}$ Hold Time	50		50		
7	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	0	500	0	500	
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		0		
9	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay	0		0		
10	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handshake	0	400	0	400	
11	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑)	0		0		
12	TdACKr(DAV)	$\overline{ACKIN}$ ↑ to $\overline{DAV}$ ↓	0	800	0	800	
13	TdACKf(Empty)	$\overline{ACKIN}$ ↓ to Empty	0		0		
14	TdACKf(Full)	$\overline{ACKIN}$ ↓ to Full	0		0		
15	TcACK	$\overline{ACKIN}$ Cycle Time	1		1		1

NOTES:
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns), except as noted.
1. Units in microseconds.

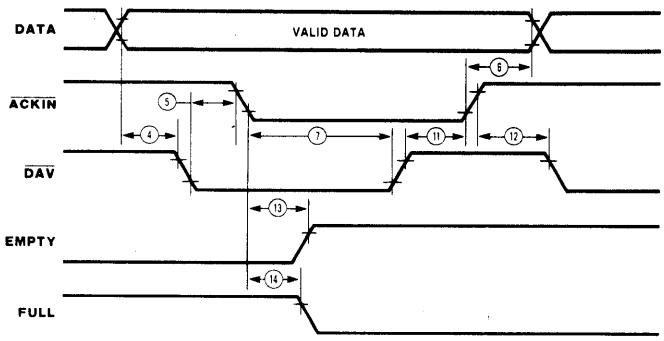


Figure 48. 2-Wire Handshake (Port 2 Side Only) Output

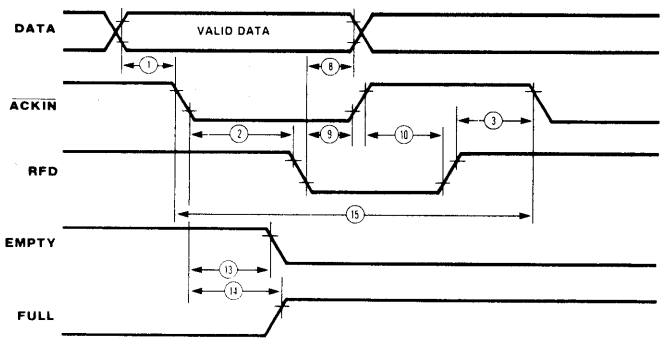


Figure 49. 2-Wire Handshake (Port 2 Side Only) Input

AC Characteristics

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsDI(DAV)	Data Input to $\overline{DAV}$ ↓ Setup Time	50		50		
2	TdDAVIf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0	500	0	500	
3	TdDAVf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0	500	0	500	
4	ThDI(DAC)	Data In to DAC ↑ Hold Time	0		0		
5	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay	0		0		
6	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0	500	0	500	
7	TdDAVIr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0	500	0	500	
8	TdRFDI(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0		0		
9	TsDO(DAC)	Data Out to $\overline{DAV}$ ↓					
10	TdDAVOf(RFD)	$\overline{DAV}$ ↓ to RFD ↓ Delay	0		0		
11	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay	0		0		
12	ThDO(DAC)	Data Out to DAC ↑ Hold Time					
13	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay		400		400	
14	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay	0		0		
15	TdDAVOr(RFD)	$\overline{DAV}$ ↑ to RFD ↑ Delay	0		0		
16	TdRFDO(DAV)	RFD ↑ to $\overline{DAV}$ ↓ Delay	0	800	0	800	

NOTES:
* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

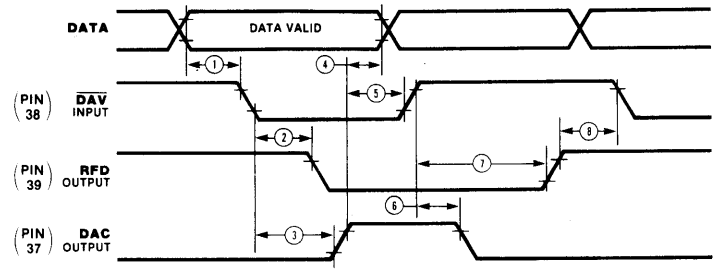


Figure 50. 3-Wire Handshake Input

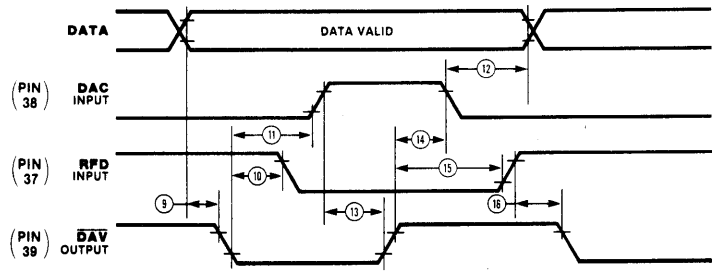


Figure 51. 3-Wire Handshake Output

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8038	CE	4.0 MHz	Z-FIO (40-pin)
Z8038	CM	4.0 MHz	Same as above
Z8038	CMB	4.0 MHz	Same as above
Z8038	CS	4.0 MHz	Same as above
Z8038	DE	4.0 MHz	Same as above
Z8038	DS	4.0 MHz	Same as above
Z8038	PE	4.0 MHz	Same as above
Z8038	PS	4.0 MHz	Same as above

Product Number	Package/ Temp	Speed	Description
Z8038A	CM	6.0 MHz	Z-FIO (40-pin)
Z8038A	CMB	6.0 MHz	Same as above
Z8038A	CS	6.0 MHz	Same as above
Z8038A	DE	6.0 MHz	Same as above
Z8038A	DS	6.0 MHz	Same as above
Z8038A	PE	6.0 MHz	Same as above
Z8038A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = 55°C to +125°C, MB = -55°C to 125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8060
Z8000™ Z-FIFO Buffer Unit and Z-FIO Expander

Zilog

Product Specification

September 1983

Features

- Bidirectional, asynchronous data transfer capability
- Large 128-bit-by-8-bit buffer memory
- Two-wire, interlocked handshake protocol
- Wire-ORing of empty and full outputs for sensing of multiple-unit buffers
- 3-state data outputs
- Connects any number of FIFOs in series to form buffer of any desired length
- Connects any number of FIFOs in parallel to form buffer of any desired width

General Description

The Z8060 First-In First-Out (Z-FIFO) Buffer Unit consists of a 128-bit-by-8-bit memory, bidirectional data transfer and handshake logic. The structure of the Z-FIFO unit is similar to that of other available buffer units. Z-FIFO is a general-purpose unit; its handshake logic is compatible with that of other members of Zilog's Z8 and Z8000 Families.

Z-FIFOs can be cascaded end-to-end without limit to form a parallel 8-bit buffer of any desired length (in 128-byte increments). Any number of single- or multiple-unit Z-FIFO serial buffers can be connected in parallel to form buffers of any desired width (in 8-bit increments).

The Z-FIFO buffer units are available as 28-pin packages. Figures 1 and 2 show the pin functions and pin assignments, respectively, of the Z-FIFO device. A block diagram is shown in Figure 3.

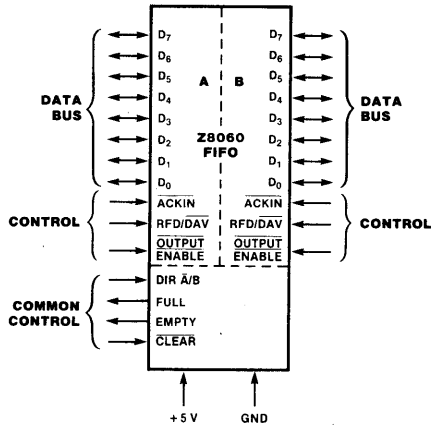


Figure 1. FIFO Pin Functions

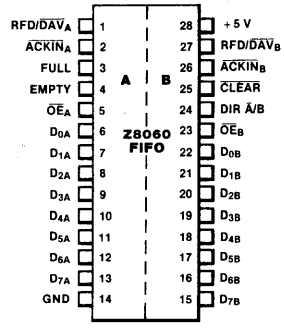


Figure 2. FIFO Pin Assignments

General Description (Continued)

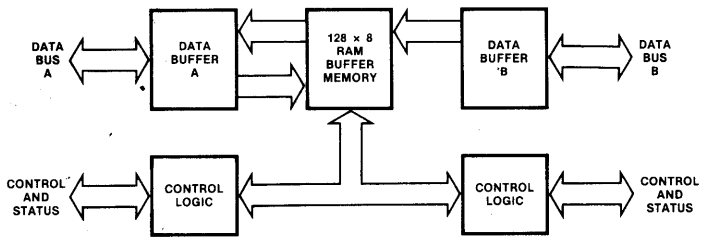


Figure 3. Functional Block Diagram

Pin Descriptions

$\overline{\text{ACKIN}}$. *Acknowledge Input* (input, active Low). This line signals the FIFO that output data has been received by peripherals or that input data is valid.

$\overline{\text{CLEAR}}$. *Clear Buffer* (input, active Low). When set to Low, this line causes all data to be cleared from the FIFO buffer.

D_0-D_7. *Data Bus* (inputs/outputs, bidirectional). These bidirectional lines are used by the FIFO to receive and to transmit data.

DIR $\overline{\text{A}}$/B. *Direction Input A/B* (input, two control states). A High on this line signals that input data is to be received at Port B. A Low on this line signals that input data is to be received at Port A.

EMPTY. *Buffer Status* (output, active High, open-drain). A High on this line indicates that the FIFO buffer is empty.

FULL. *Buffer Status* (output, active High, open-drain). A High on this line indicates that the FIFO buffer is full.

$\overline{\text{OEA}}$, $\overline{\text{OEB}}$. *Output Enable A, Output Enable B* (inputs, active Low). When Low, $\overline{\text{OEA}}$ enables the bus drivers for Port A; when High, $\overline{\text{OEA}}$ causes the bus drivers to float to a high-impedance level. Input $\overline{\text{OEB}}$ controls the bus drivers for Port B in the same manner as OEA controls those for Port A.

RFD/$\overline{\text{DAV}}$. *Ready-for-Data/Data Available* (outputs RFD, active High; $\overline{\text{DAV}}$ active Low). RFD, when High, signals to the peripherals involved that the FIFO is ready to receive data. $\overline{\text{DAV}}$, when Low, signals to the peripherals involved that FIFO has data available to send.

Functional Description

Interlocked 2-Wire Handshake. In interlocked 2-wire handshake operation, the action of FIFO must be acknowledged by the other half of the handshake before the next action can occur. In an Output Handshake mode, the FIFO indicates that new data is available only after the external device has indicated that it is ready for the data. In an Input Handshake mode, the FIFO does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the acceptance of the last byte. This control feature allows the FIFO, with no external logic, to directly interface with the port of any CPU in the Z8 Family—a CIO, a UPC, an FIO, or another FIFO. The timing for the input and output handshake operations is shown in Figures 4 and 5, respectively.

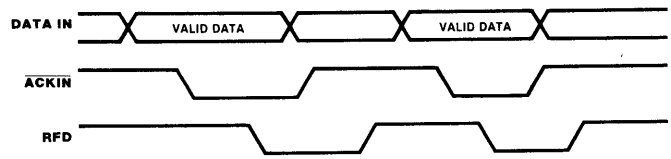


Figure 4. Two-Wire Interlocked Handshake Timing (input)

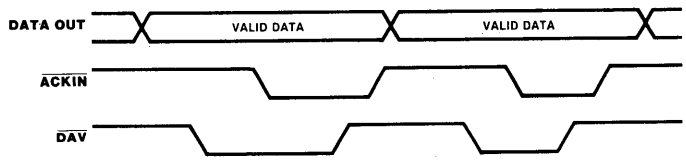


Figure 5. Two-Wire Interlocked Handshake Timing (output)

Functional Description (Continued)

Resetting or Clearing the FIFO. The $\overline{\text{CLEAR}}$ input is used to initialize and clear the FIFO. A Low level on this input clears all data from the FIFO, allows the EMPTY output to go High and forces both outputs RFD/$\overline{\text{DAV}}_A$ and RFD/$\overline{\text{DAV}}_B$ High. A High level on $\overline{\text{CLEAR}}$ allows the data to transfer through the FIFO.

Bidirectional Transfer Control. The FIFO has bidirectional data transfer capability under control of the DIR $\overline{\text{A}}$/B input. When DIR $\overline{\text{A}}$/B is set Low, Port $\overline{\text{A}}$ becomes input handshake and Port B becomes output handshake; data transfers are then made from Port A to Port B. Setting DIR $\overline{\text{A}}$/B High reverses the handshake assignments and the direction of transfer. This bidirectional control is illustrated in Table 1.

DIR $\overline{\text{A}}$/B	Port A Handshake	Port B Handshake	Transfer
0	Input	Output	A to B
1	Output	Input	B to A

Table 1. Bidirectional Control Function Table

The FIFO buffer must be empty before the direction of transfer is changed; otherwise, the results of the change will be unpredictable. If FIFO status is unknown when a transfer direction change is to be made, the recommended procedure is:

(1) Force and hold $\overline{\text{CLEAR}}$ Low.

(2) Set DIR $\overline{\text{A}}$/B to the level required for the desired direction.

(3) Force $\overline{\text{CLEAR}}$ High.

Empty and Full Operation. The EMPTY and FULL output lines can be wire-ORed with the EMPTY and FULL lines of other FIFOs and FIOs. This capability enables the user to determine the empty/full status of a buffer consisting of multiple FIFOs, FIOs, or a combination of both. Table 2 shows the various states of EMPTY and FULL.

Number of Bytes in FIFO	EMPTY	FULL
0	High	Low
1-127	Low	Low
128	Low	High

Table 2. Signals EMPTY and FULL Operation Table

Interconnection Example. Figure 6 illustrates a simplified block diagram showing the manner in which FIFOs can be interconnected to extend a FIO buffer.

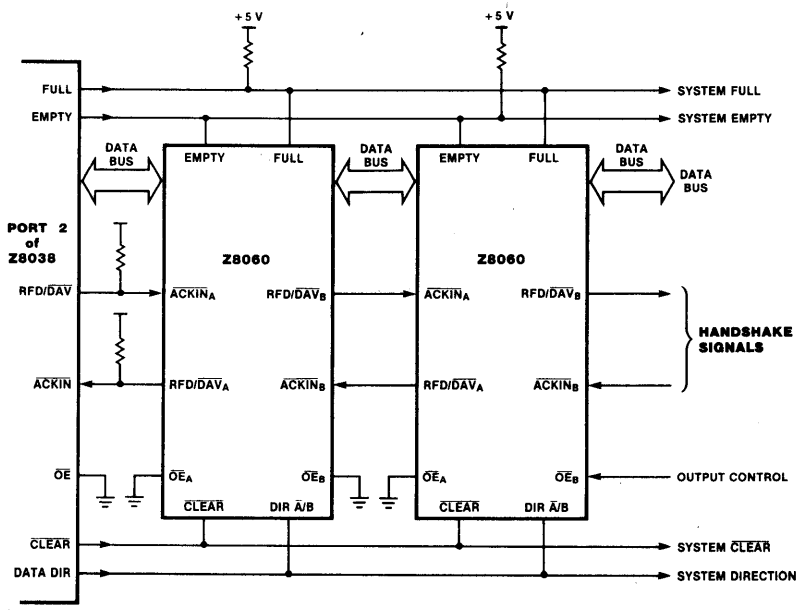


Figure 6. Typical Interconnection (Simplified Diagram)

Functional Description (Continued)

Output Enable Operation. The FIFO provides a separate Output Enable ($\overline{OE}$) signal for each port of the buffer. An $\overline{OE}$ output is valid only when its port is in the Output Handshake mode. The control of this output function is shown in Table 3. Signal $\overline{OE}$ operates with lines DIR $\overline{A}$/B. A High on a valid $\overline{OE}$ line 3-states its port's data bus but does not affect the handshake operation. A Low level on a valid $\overline{OE}$ enables the data bus outputs if its port is in the Output Handshake mode. Note that the handshake operation is unaffected by the Output Enable pin.

DIR $\overline{A}$/B	$\overline{OE}_A$	$\overline{OE}_B$	Function
0	X	0	Disable Port A Output Enable Port B Output
0	X	1	Disable Port A Output Disable Port B Output
1	0	X	Enable Port A Output Disable Port B Output
1	1	X	Disable Port A Output *Disable Port B Output*

NOTE: X = Don't care.

Table 3. Output Control Function Table

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND .....-0.3 V to +7.0 V
Operating
Ambient Temperature.....As specified in Ordering Information
Storage Temperature.....-65° to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information. All ac parameters assume a load capacitance of 50 pF max.

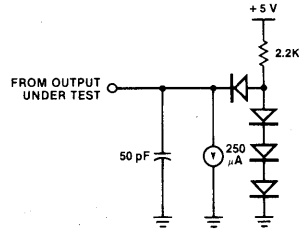


Figure 7. Standard Test Load

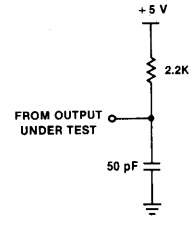


Figure 8. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IH}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 A
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
			0.5	V	I_{OL} = +3.2 mA
I_{IL}	Input Leakage		±10	μA	$0.4 \leq V_{IN} \leq +2.4$ V
I_{OL}	Output Leakage		±10	μA	$0.4 \leq V_{OUT} \leq +2.4$ V
I_{CC}	V_{CC} Supply Current		200	mA	

NOTE: V_{CC} = +5 V ±5% unless otherwise specified over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured pins
C_{OUT}	Output Capacitance		15	pF	returned to ground
$C_{I/O}$	Bidirectional Capacitance		20	pF	
Input					
tr	Any input rise time		100	ns	
tf	Any input fall time		100	ns	

NOTE: f = 1 MHz over specified temperature range.

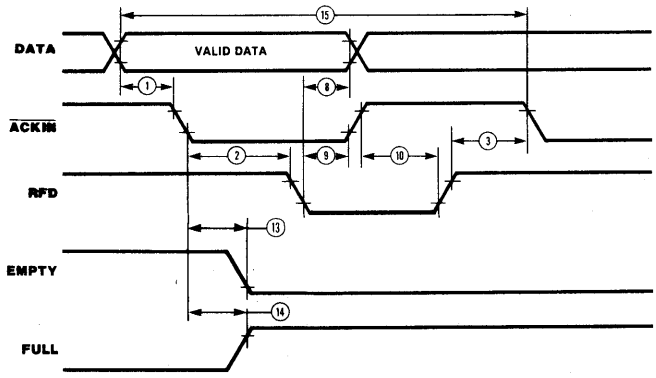
Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8060	CE	4.0 MHz	FIFO (28-pin)	Z8060	DS	4.0 MHz	FIFO (28-pin)
Z8060	CS	4.0 MHz	Same as above	Z8060	PE	4.0 MHz	Same as above
Z8060	DE	4.0 MHz	Same as above	Z8060	PS	4.0 MHz	Same as above

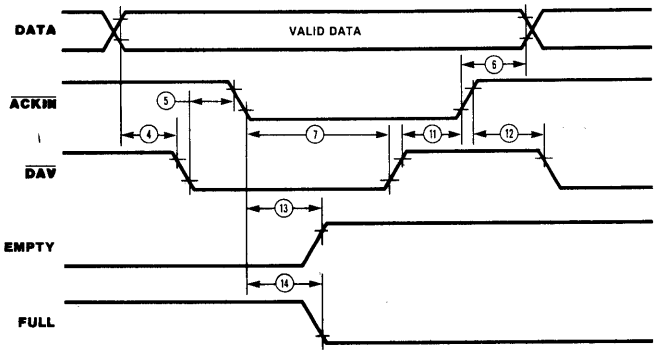
NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, S = 0°C to 70°C

2-Wire Interlocked Handshake Timing

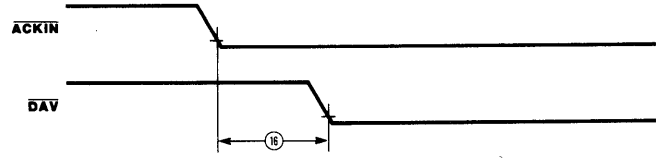
INPUT TIMING



OUTPUT TIMING



ACKNOWLEDGE INPUT TO DATA AVAILABLE TIME (BUBBLE TIME)



OUTPUT ENABLE AND CLEAR

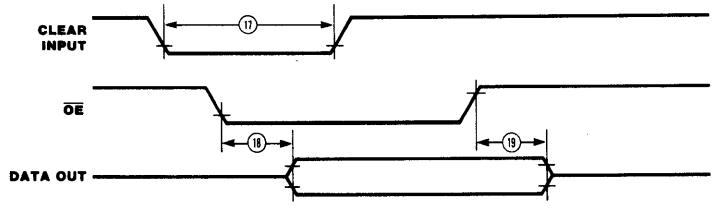


Figure 9. Timing Diagrams

FIFO 2-Wire Handshake Timing. Timing for 2-wire interlocked handshake operation is shown in Figure 9. The symbol, description and values for the numbered parameters (Figure 9) are given in AC Characteristics.

AC Characteristics

No.	Symbol	Parameter	Min	Max	Units*
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ to Setup Time	50		ns
2	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		ns
3	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		ns
4	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	50		ns
5	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		ns
6	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↑ Hold Time	50		ns
7	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	0		ns
8	ThDI(RFD)	Data Input to RFD ↓ Hold Time	0		ns
9	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay	0		ns
10	TdACKr(RFD)	$\overline{ACKIN}$ ↑ to RFD ↑ Delay	0		ns
11	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑	0		ns
12	TdACKr(DAV)	$\overline{ACKIN}$ ↑ to $\overline{DAV}$ ↓	0		ns
13	TdACKINf(EMPTY)	(Input) $\overline{ACKIN}$ ↓ to EMPTY ↓ Delay (Output) $\overline{ACKIN}$ ↓ to EMPTY ↑ Delay			
14	TdACKINf(FULL)	(Input) $\overline{ACKIN}$ ↓ to FULL ↑ Delay (Output) $\overline{ACKIN}$ ↓ to FULL ↓ Delay			
15	ACKIN Clock Rate	(Input or Output)	1.0		MHz
16	TdACKINf(DAVf)	(Bubble Time)			ns
17	TwCLR	Width of Clear to Reset FIFO	700		ns
18	TdOE(DO)	$\overline{OE}$ ↓ to Data Bus Driven	0		ns
19	TdOE(DRZ)	$\overline{OE}$ ↑ to Data Bus Float			ns

NOTES:
* All timing references assume 2.0 V for a logic 1 and 0.8 V for a logic 0. Timings are preliminary and subject to change.

Z8065
Z8000™ Z-BEP
Burst Error Processor

Zilog

Product Specification

September 1983

Features

- Detects errors in serial data up to 585,442 bits in length
- Implements correction of a detected error burst of up to 12 bits in length
- Handles effective data rates of up to 20M bits per second
- Provides four industry-standard polynomials for error detection
- Designed for use in both microprocessor and microprogrammed disk-controlled systems
- Provides three correction algorithms:
  - Full-period clock-around method for conformance to current practices
  - Chinese remainder theorem, which reduces correction time by orders of magnitude
  - Reciprocal polynomial, which allows correction with 48-bit code

General Description

The Z8065 Burst Error Processor (BEP) provides error detection and correction facilities for high-performance, high-density disk systems and any other system in which high-speed serial data transfers occur.

For error detection, the BEP provides a selection of four standard polynomials, including the more popular 56-bit and 48-bit versions, to satisfy a broad range of applications. During write operations, the BEP generates check-bit words, which are appended to the record being written onto the disk. These check-bit words are then used in subsequent reads and, if necessary, in correction operations.

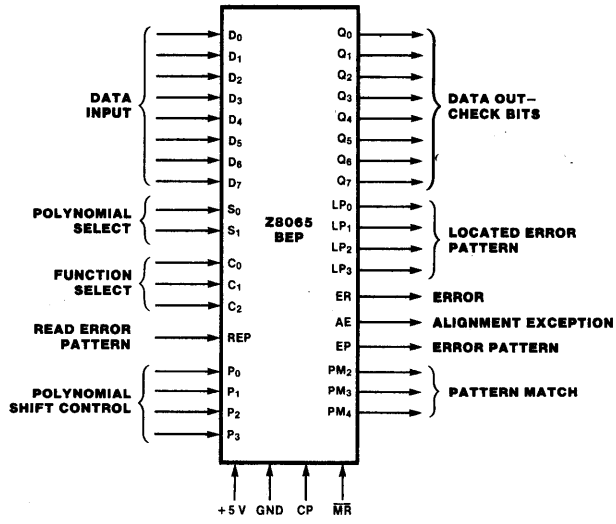


Figure 1. Pin Functions

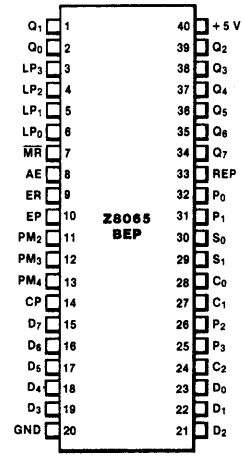


Figure 2. Pin Assignments

2019-001 2019-002 Some of the material used herein is used by permission of Advanced Micro Devices, Inc.

General Description (Continued)

When a stored record is read, the BEP computes the syndrome for data validation. This syndrome is then used to determine if an error burst is present in the retrieved data stream. If an error is detected, the BEP can be used to locate its actual bit pattern. The information obtained is then made available to the host system where it is used to correct the data read. Any of the three algorithms can be selected for this process.

The BEP is fabricated using silicon-gate, N-MOS technology and is supplied in a 40-pin dual in-line package (DIP). Figures 1 and 2 illustrate the pin functions and pin assignments, respectively, of the Z8065. Z8065 operation requires a +5 V dc power supply and a single-phase clock.

Pin Descriptions

AE. *Alignment Exception* (output, active High). This output goes High when a misalignment condition occurs during an error-pattern search operation.

C_0-C_2. *Function Select* (inputs, active High). These three lines carry the binary code used to select the BEP functions. The codes and the functions initiated by each are listed in Table 1.

C_2	C_1	C_0	Function
L	L	L	Compute Check Bits
L	L	H	Write Check Bits
L	H	L	Read Normal
L	H	H	Read High Speed
H	L	L	Load
H	L	H	Reserved
H	H	L	Correct Normal (Full Period Clock Around)
H	H	H	Correct High Speed (Chinese Remainder Theorem Method)

H = High, L = Low

Table 1. Function Select Codes

CP. *Clock* (input, active High). All BEP operations are timed by this external clock input. Any input changes must be made to the BEP when CP is High. Data is strobed into the BEP only during the Low-to-High transition of CP. BEP outputs are valid only after a subsequent Low-to-High transition occurs on CP.

D_0-D_7. *Data In* (input, active High). These eight lines are used to enter data into the BEP. D_0 is the least-significant bit (LSB) position of the input; D_7 is the most-significant bit (MSB) position of the input. Data entry occurs on the Low-to-High transition of the input clock pulse. Any change on D_0-D_7 must take place when the clock pulse (CP) input is High.

EP. *Error Pattern* (output, active High). During an error correction process, EP is set High to indicate that the bit pattern of the detected error has been found. EP is set Low each time the BEP is initialized. EP is valid only during the performance of a correction function; it must be ignored at all other times.

ER. *Error* (output, active High). ER indicates that the BEP has detected an error in the input data stream. If the register array contains a zero syndrome, ER is set Low to indicate that no error was detected. If the array contains a non-zero syndrome, ER is set High to indicate that an error was detected. The output is valid only after the BEP receives the last check byte during a normal read or a read high-speed function. The resulting syndrome is contained by the register array. ER is set Low each time the BEP is initialized.

LP_0-LP_3. *Located Error Pattern* (outputs, 3-state). These four lines, together with Q_0-Q_7 provide a 12-bit error pattern that is output when REP is High. Q_7 is the MSB of the pattern; LP_0 is the LSB of the pattern. A High level on any output line represents a logical 1; a Low level a logical 0. When no error pattern is available (REP is Low), output lines LP_0-LP_3 and Q_0-Q_7 are maintained in a high-impedance state.

$\overline{MR}$. *Master Reset* (input, active Low). This input controls the initialization of the BEP. Setting $\overline{MR}$ Low for a minimum period of 800 ns initialized the BEP. The BEP must be initialized prior to performing compute check bits, read normal, read high-speed, and load functions.

P_0-P_3. *Polynomial Shift Control* (inputs, active High). During correction procedures using the Chinese remainder theorem, each syndrome obtained by the high-speed read function is shifted individually. The P_0-P_3 inputs provide this capability: P_0 enables the shifting of the first syndrome, P_1 shifts the second syndrome and so on. A High on an input allows the corresponding register to shift; a Low causes it to hold. These inputs are effective only during the correct high-speed function. Changes on these inputs occur only when the CP input is High.

PM_2-PM_4. *Pattern Match* (outputs, active High). These lines are used during a Chinese remainder theorem error-correction operation to indicate error-pattern match conditions for each syndrome involved. When High, an output specifies that the corresponding syndrome register has achieved a match.

Q_0-Q_7. *Data Out* (outputs, 3-state). These eight lines are active only during write check bit and error correction functions. At all other times, Q_0-Q_7 are maintained at a high-impedance level. During the write check bit

Pin Descriptions (Continued)

function, check bits are presented to these lines one byte at a time. Q_0 is the LSB and Q_7 is the MSB of the output. During the error-correction function, REP enables these lines to carry the detected error bit pattern.

REP. *Read Error Pattern* (input, 3-state). REP when High, enables lines LP_0-LP_3 and Q_0-Q_7. This error pattern information is valid only after a High is indicated on the EP output during correction operations.

S_0-S_1. *Polynomial Select* (inputs, active High). These two pins carry the binary codes required to select which polynomial the BEP will implement. The select codes (logic levels) are given in Table 2.

S_1	S_0	Polynomial	Number of Check Bits
L	L	$(X^{22}+1)(X^{11}+X^7+X^6+X+1)(X^{12}+X^{11}+X^{10}+X^9+X^8+X^7+X^6+X^5+X^4+X^3+X^2+X+1)$ $(X^{11}+X^9+X^7+X^6+X^5+X+1)$	56
L	H	$(X^{21}+1)(X^{11}+X^2+1)$	32
H	L	$(X^{23}+1)(X^{12}+X^{11}+X^8+X^7+X^3+X+1)$	35
H	H	$(X^{13}+1)(X^{35}+X^{23}+X^8+X^2+1)$	48

H = High, L = Low

Table 2. Polynomial Select Codes

Architecture

The BEP consists of four major circuit groups: control logic, polynomial divide matrix, register array, and status logic. Figure 3 shows a block diagram of the BEP.

Control Logic. The control logic circuits provide timing, reset, polynomial selection, and read error-pattern control inputs for the remaining BEP circuits.

Basic timing is provided to the control logic by clock input CP. The control logic generates and distributes appropriate timing and control signals to the remaining BEP circuits.

Enabling the Master Reset ($\overline{MR}$) causes the control logic to initialize all device circuits. This operation is usually performed before the execution of a selected device function.

Function select and polynomial select inputs are decoded by the control logic. The outputs of the decoder are then used to generate the control and timing signals needed to perform the encoded function or to select the encoded polynomial.

The Read Error Pattern (REP) and Polynomial Shift Control signals enable the control logic to strobe valid error bit pattern outputs onto the register array output lines. During high-speed corrections, the polynomial shift inputs are used for register array, error burst-pattern bit-matching operations.

Polynomial Divide Matrix. This matrix connects the register array circuits so that each data byte presented on lines D_0-D_7 is suitably divided by the user-selected polynomial. The connections to be made are determined by

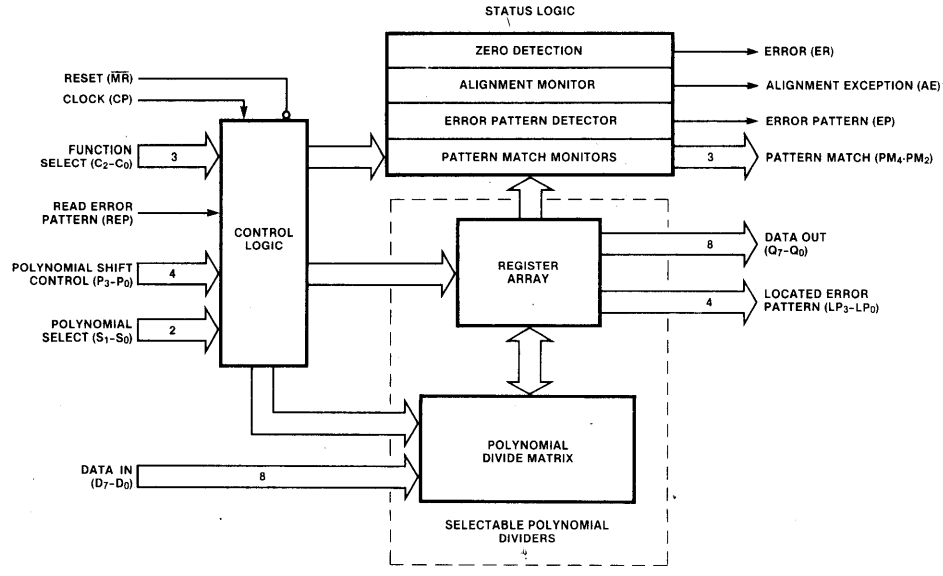


Figure 3. Simplified Block Diagram

Architecture (Continued)

gating the signals supplied by the control logic after decoding the select code on inputs S_0 and S_1.

Register Array. This array consists of 56 flip-flop circuits used for: (1) check-bit computation during write operations, (2) syndrome computation during read operations, and (3) error pattern extraction during error-correction operations.

The bit patterns required for array functions are provided by the polynomial divide matrix. The array and matrix circuits, together, simulate a serial, polynomial, feedback-shift register arrangement in an 8-bit parallel form.

At the end of each write operation, the computed check-bit bytes are available on lines Q_0-Q_7. On completion of a correction operation, the bit pattern of the detected error is available on lines LP_0-LP_3 and Q_0-Q_7. Input REP determines when a valid error bit pattern is on the register output lines.

Status Logic. These circuits monitor the register array to detect the conditions listed in Table 3 and to enable the generation of the corresponding control signals.

Condition Detected	Signal
Result of Polynominal Division	ER (error) High when error found; Low when no error.
Alignment during H-S and normal correction	AE (alignment error) High when error pattern is incorrectly aligned.
Location of an error bit pattern	EP (error pattern) High when an error-bit pattern is detected.
Pattern matching during H-S correction	PM_2, PM_3, PM_4 (pattern match). Each signal goes High when its corresponding register matches the proper section of a located burst-bit pattern.

Table 3. Status Logic, Detected Condition and Resulting Output

Functional Description

The BEP detects and corrects data errors using write, read, and correct operations. The BEP operates in conjunction with external logic: either a microprocessor or microprogrammed control circuitry. Master clock inputs, the selection of polynomials and functions, and the output of check bytes and error bit patterns are initiated and controlled by inputs from external circuits. External logic is also used to collect data, perform calculations, and carry out the actual modification of stored data during error-correction operations.

The BEP contains code for four standard polynomials (sometimes referred to as Fire codes). This code forms the basis fo the unit's error detection and correction functions. The polynomial to be used is selected by a coded input from the host system*. Table 2 lists the polynomial select codes, the equations implemented, and the number of check bits generated by each polynomial during a write operation. The same polynomial must be selected for the write, read, and correction operations performed for a given data stream. It is the responsibility of the host system to keep track of which polynomial is selected for use with each data stream.

The BEP also contains the code required to implement each of seven functions that can be executed during data stream write, read, and correction operations. The function to be performed is selected by a coded input from the host system.* The functions and their required input code are listed in Table 1.

*NOTE: In the remainder of this specification, external circuitry and software is referred to as the host system.

Write Operation. Before data is written onto a disk or similar storage device, the BEP must generate and add check-bit bytes to the data to be stored. These bytes are required for the detection and correction of errors that may occur during write and during subsequent read operations.

Immediately before a write operation, the host system must output codes to the BEP to select the polynomial to be used and to initiate the compute check-bit function.

The data stream to be written is entered, on-the-fly, into the BEP as it is written onto the disk. Data is presented to the BEP as a series of 8-bit bytes in parallel form. Check bits are generated by dividing each input byte by the selected polynomial using the rules of algebra in polynomial fields. The check bits are stored as they are generated. Check bits are generated in sets of 56, 48, 35, or 32 bits, depending on the selected polynomial. When the last input byte has been processed, the write check-bit function is initiated by the host system. During a check-bit write, the generated check bits are organized into 8-bit bytes (check-bit bytes), which are output byte by byte in 8-bit parallel form on lines Q_0-Q_7. The host system adds these check-bit bytes to the end of the newly written file.

The polynomial selected for data write operations must also be used in subsequent reads of the written data. Therefore, in selecting a polynomial for a write operation, the user should consider the type of read and correction functions desired for future data retrieval operations. The relationships between

Functional Description (Continued)

the polynomials and the read and corrections functions are:

- A read normal function must be followed by a correct normal function. All four polynomials can be used with this set of read/correction functions.
- A read high-speed function must be followed by the Chinese remainder theorem correction function. All but the 48-bit polynomial can be used with this set of read/correction functions.

Read Operations. When data is read from a disk, the BEP checks the retrieved data and check-bit bytes for read and write errors. If errors are detected, the host system initiates correction functions, retrieving from the BEP the information needed to locate and correct the erroneous data. Immediately before starting the read operation, the host system selects the desired polynomial and either a read normal or a read high-speed function. Data and associated check-bit bytes are then loaded, byte-by-byte, into the BEP as they are read from the disk. A divide operation results in one or more syndromes (depending on the polynomial used), which are stored in the register array. The binary values of these syndromes indicate whether or not an error was present in the scanned data stream. If an error was detected, ER is set High.

When an error condition is indicated and correction is desired, the host system must initiate a correct normal, a 48-bit correct normal, or a correct high-speed function. The function selected depends on which polynomial and which read function were selected for the initial read operation. When executed, the selected correction function supplies the host system with the information needed to calculate the location of the error pattern in the data stream and to correct the erroneous data stream bits.

Read Normal Function. When this function is selected, the polynomial matrix is configured to establish the selected polynomial in its expanded form. The input stream, data and check bit bytes, are divided byte by byte by the expanded polynomial. The results form a syndrome whose binary value is detected by the status logic. If the syndrome is nonzero, ER goes High, to indicate an error condition; if the syndrome is zero, ER remains Low, to indicate a no-error state.

Read High-Speed Function. This function is selected when the correct high-speed function is used. All but the 48-bit polynomial can be used with this function.

When selected, this function configures the polynomial matrix to simultaneously divide each byte of the input data/check-bit stream by all factors of the selected polynomial. The result of each factor division forms a separate syndrome. Thus, the number of syndromes developed depends upon the number of factors in the selected polynomial. The status logic monitors all syndromes and uses their combined binary values to determine if an error condition is present. If all syndromes are zero after the last byte of the input stream is read, a no-error state is indicated and ER remains Low. If any syndrome has a non-zero value, an error condition is indicated and ER is set High.

48-Bit Polynomial. Only read normal and correct normal functions can be used when this polynomial is selected. The read normal function for the 48-bit polynomial is performed in the same manner as for the other polynomials. The resulting syndrome, however, will be too long and cannot be used directly in subsequent correct normal functions; instead, the reciprocal of the syndrome must be established in the BEP before the correct normal function is selected. The host system initiates this operation by selecting the write check-bit function immediately after the syndrome is formed and error is indicated. Clock pulses are then applied to the BEP during the write bit function to strobe the syndrome onto lines Q_0-Q_7 as six sequential 8-bit bytes. The host system must then reverse the order of the syndrome bits (that is, the original LSB becomes the new MSB and the original MSB becomes the new LSB) to form the reciprocal. The host system then reloads this new syndrome into the BEP by selecting the load function.

Load Function. This function is used only during read normal and correct normal operations when the 48-bit polynomial is selected. The host system selects the load function to prepare the BEP to receive an externally-formed reciprocal syndrome and to control the loading of the syndrome bytes into the BEP. The load function causes the register array to be configured into an 8-bit wide, 7-bit deep shift register connected to lines D_0-D_7. The host system then presents the six syndrome bytes on lines D_0-D_7. Clock pulses are then generated to strobe the bytes into the Shift register one at a time.

When all six bytes of the syndrome are loaded, the host system causes the input lines to be pulled Low, then generates a seventh clock pulse. The seventh clock pulse strobes these Lows into the Shift register as a zero dummy fill byte. On completion of the load operation, the BEP is ready for the correct normal function.

Correction Operations

The detection of an error in a retrieved data stream causes the following corrective functions to be performed:

1. The error burst containing the erroneous data bits is located by the BEP.
2. Using data supplied by the BEP, the host system calculates the exact position of the error burst in the retrieved data stream.
3. The bit pattern of the error burst is strobed out of the BEP and used by the host system to perform bit correction.

Location of an Error Pattern. An error pattern is characterized by the appearance of a known number of consecutive 0s in specific registers of the register array. The exact number of 0s and their locations in the register array is unique to each polynomial. When a polynomial is selected for read and error-detect/correction operations, the pattern associated with that polynomial is loaded into the status logic. The status logic uses this pattern to identify an error burst during the error pattern location operations.

When only one syndrome is developed during the read error-detection function, the error-bit pattern is located by repeatedly dividing the syndrome by the polynomial. Division is accomplished by the repeated application of clock pulses (CP), while ignoring the states of lines D_0–D_7. This operation results in a serial bit-by-bit reconstruction of the retrieved data stream. The generated data bits are shifted at a rate of one per clock cycle through the register array. The BEP status logic performs the actual error bit pattern detection as the data stream is reconstructed. The status logic monitors specific registers of the register array, and it detects a pattern of 0s that matches the zero error pattern unique to the selected polynomial, it sets EP High to indicate that an error burst was found.

When more than one syndrome is developed during the read error-detection function, each syndrome is divided by its associated factor until a match condition is found for each. Each time a match is found, the status logic enables one of outputs PM_2, PM_3 or PM_4. When the total error bit pattern is found, the status logic outputs associated with the syndromes of a polynomial (2 or 4) are all enabled. The clock pulses required for each factor syndrome divide operation are supplied by lines P_0–P_3.

A major factor in calculating the exact location of error-burst patterns in the retrieved data stream is the number of clock pulses used by the BEP to detect the error-burst pattern. The host system must record the total number of clock pulses generated from the start of BEP pattern-location operations to the enabling of output EP. If necessary, this total must include the clock pulses needed for alignment operations.

Bit Alignment. During syndrome division, the register array is configured into a matrix representing an 8-bit, parallel mechanization of a serial, polynomial division scheme. Under certain conditions the error pattern bits developed do not line up automatically. When the status logic detects such a misalignment, AE is set High. When AE is High, the BEP switches internally into One-Bit Shift mode during which each input clock pulse shifts the data stream one cell through the register array. When alignment is achieved, the status logic sets AE Low and the BEP is switched out of the One-Bit Shift mode. The number of shift pulses needed to achieve alignment is an additional factor in calculating the position of the error-bit pattern.

Uncorrectable Errors. If the total clock cycle time needed to locate the error burst pattern is greater than the natural period of the selected polynomial (Table 4), an uncorrectable error condition is indicated and the host system must abort the correction operation.

Polynomial	No. of Check Bits	Period (Bits)	Correctable Burst Error Length (Bits)
$(X^{22}+1)(X^{11}+X^7+X^6+X+1)(X^{12}+X^{11}+X^{10}+\ldots$ $+X+1)(X^{11}+X^9+X^7+X^6+X^5+X+1)$	56	585,442	11
$(X^{21}+1)(X^{11}+X^2+1)$	32	42,987	11
$(X^{23}+1)(X^{12}+X^{11}+X^8+X^7+X^3+X+1)$	35	94,185	12
$(X^{13}+1)(X^{35}+X^{23}+X^8+X^2+1)$	48	$13(2^{35}-1)$	7

Table 4. Polynomials, Checkbits, Natural Period, and Length of Error Burst

Correction Operations (Continued)

Correct Normal Function. This function must be preceded by a read normal function. With the exception of 48-bit polynomial operations, this function performs all operations needed to construct a serial form of the retrieved data stream and to locate the detected error burst. The operations performed are the same as those described previously for a single-syndrome situation.

Computing Error Bit Pattern Locations. *If no* alignment exception state is indicated (AE is Low), the locations of the error-bit pattern within the data stream (except when the 35-bit polynomial is used) can be calculated by the formula:

$$L = NK - 8R1$$

Where:

L = The location (number) of the first bit in the error burst, counting from the last check bit in the scanned record.

N = The natural period of the selected polynomial.

K = The smallest integer needed to make this expression positive.

R1 = The total number of clock pulses input by the BEP from the start of the find operation until EP goes High.

If an alignment exception state is indicated (AE is High), the location of the error-bit pattern within the data stream (except when the 35-bit polynomial is used) can be calculated using the formula:

$$L = NK - 8R1 - R2$$

Where:

L,N,K, and R1 are the same as described above.

R2 = The number of clock pulses input by the BEP between the time that AE goes High and EP goes High.

If the 35-bit polynomial is selected, the quantity 5 must be added in the following manner to the formulas used to calculate the location of the error-bit pattern:

$$L = NK - 8R1 + 5$$

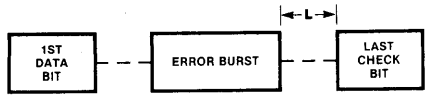
$$\text{and } L = NK - 8R1 - R2 + 5$$

Correct Normal Function, 48-Bit Polynomial. The functions performed by the correct normal function when a 48-bit polynomial is selected are essentially the same as those described for the other polynomials. The major exception is that the location of the first bit in the error-bit pattern is calculated using the formula:

$$L = (8R1 + R2) - 48$$

Where:

L = The number of bit positions from the last check bit to the nearest error burst bit.



R1 = If alignment is needed, R1 is the number of clock pulses from the start of the find operation until AE goes High. If no alignment is needed, R1 is the total number of clock pulses from the start of the find operation until EP goes High.

R2 = Variable used only when an alignment is needed; it represents the number of clock pulses from the time AE goes High until EP goes High.

Correct High Speed Function. This function uses a Chinese remainder theorem to locate a detected error-burst bit pattern. This theorem minimizes the number of clock pulses required for the location process, thus making it appreciably faster than the correct normal method.

The correct high-speed function must be preceded by the read high-speed function. The multiple syndromes developed during the read operation are located in consecutive sets of flip-flops in the register array. The set of flip-flops containing syndromes is treated as an individual shift register. Each syndrome shift register is associated with the factor of the polynomial used to develop the syndrome. For example, the 56-bit polynomial has four factors (see Table 2), and when selected for read and correction operations it causes four corresponding syndromes to be developed, each housed in an individual shift register in the register array.

Correction Operations (Continued)

The actual location of an error-bit pattern can be computed by the host system using the following elements:

1. The number of clock pulses required (per factor/syndrome register) to find the error pattern.
2. The natural period of each factor of the selected polynomial.
3. A predetermined constant per factor.

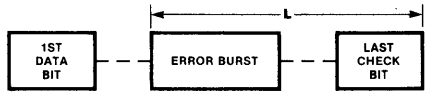
The formulas used to calculate error-pattern locations for high-speed operations are described in Table 5. Table 6 lists the predetermined constants for each factor of the polynomials. Table 7 lists the natural period of each factor of each polynomial.

56-Bit	L = (NK) - (A1M1 + A2M2 + A3M3 + A4M4)
32-Bit	L = (NK) - (A1M1 + A2M2)
35-Bit	L = (NK) -(A1M1 + A2M2 +5)

Table 5. Correct High-Speed, Error-Burst Location Formulas

Legend:

L = Beginning (first bit) of detected error burst counting from the last check bit in the processed record.



K = Smallest integer required to make right side of equation positive.

A_1-A_4 = Predetermined constants for each factor of the selected polynomial.

N = Natural period of selected polynomial.

M_1-M_4 = Number of clock pulses required to achieve a match in each factor/syndrome register.

A detected error-bit pattern can be strobed from the BEP in 12-bit, parallel form by forcing REP High when EP goes High. The 12-bit output is then matched bit for bit with the corresponding bits in the stored data. The error-bit pattern is then XORed with the matching data stream bits to effect the required bit-by-bit correction.

Figure 4 illustrates the format for strobing the error bit pattern (11 or 12 bits) out of the BEP in all but the 48-bit polynomial correction operation and shows how the pattern must be oriented to the data stream bits. Figure 5 illustrates the format for strobing the error-bit pattern (7 bits) out of the BEP in the 48-bit polynomial correction operation and shows how the pattern must be oriented to the data stream bits.

Data Stream Correction Function. Each detected error pattern consists of 12 consecutive bits not all of which represent errors. If the data and check-bit stream scanned during the preceding read operation was stored in accessible memory, the error-bit pattern can be used directly to correct the data stream.

Polynomial	Predetermined Constants A_1	A_2	A_3	A_4
56 Bit	452,387	2,521,404	578,864	2,647,216
32 Bit	311,144	32,760	—	—
35 Bit	32,760	720,728	—	—

Table 6. Chinese Remainder Theorem Coefficients

Correction Operations (Continued)

Table 7. Natural Periods for Polynomials and Polynomial Factors

Polynomial	Period Factor 1	Period Factor 2	Period Factor 3	Period Factor 4	Composite Period (n)
56 Bit	22	13	89	23	585442
32 Bit	21	2047	—	—	42987
35 Bit	23	4095	—	—	94185

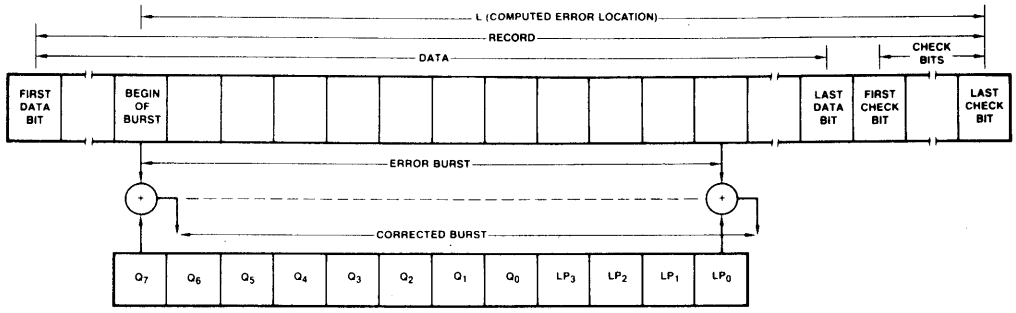


Figure 4. Error Pattern Format for 56-Bit, 35-Bit, and 32-Bit Polynomials

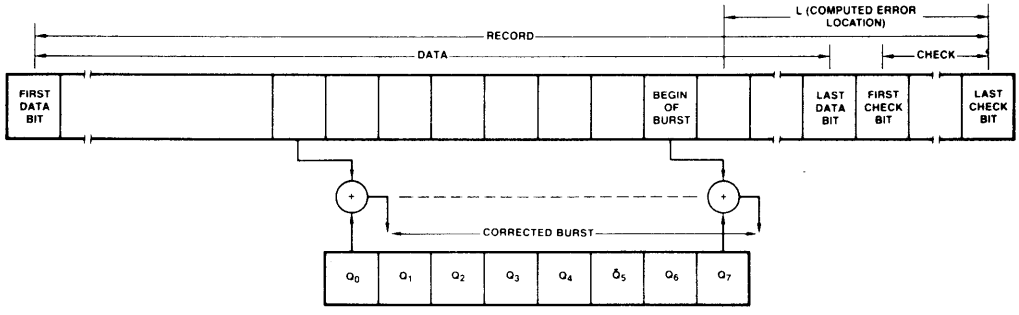


Figure 5. Error Pattern Format for 48-Bit Polynomial

Timing

The overall timing requirements for BEP operations are illustrated in Figures 6 through 13. Individual timing parameters are identified numerically in each timing diagram and are described in the AC Characteristics section.

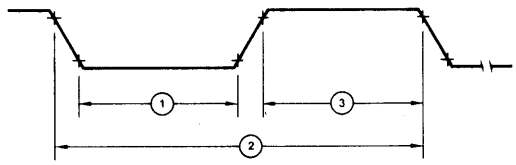


Figure 6. Clock Waveform For All Functions Except Correct Normal Or Correct High-Speed

AC Characteristics*

No.	Symbol	Parameter	Min (ns)	Max (ns)
1	TwCP1	Clock Pulse (CP) Width (Low)	180	
2	TcCP	Clock Pulse Cycle Time	400	
3	TwCPh	Clock Pulse Width (High)	180	
4	TwMR1	$\overline{MR}$ Pulse Width (Low)	800	
5	TdMR(CP)	$\overline{MR}$ ↑ to CP ↓ Time Delay—Recovery	250	
6	TsDI(CP)	DI (D_0-D_7) to CP ↑ Setup Time	350	
7	ThCP(DI)	CP ↑ to DI (D_0-D_7) Hold Time	0	
8	TsC(CP) or TsS(CP)	C(C_0-C_2) or S(S_0-S_1) to CP ↑ Setup Time	400	
9	ThCP(C) or ThCP(S)	CP ↑ to C(C_0-C_2) or S(S_0-S_1) Hold Time	0	
10	TsC(CP) or TsS(CP)	C(C_0-C_2) or S(S_0-S_1) to CP ↓ Setup Time	180	
11	TdC(Q) or TdS(Q)	C(C_0-C_2) or S(S_0-S_2) to Q(Q_0-Q_7) Valid Time Delay		200
12	TdCP(Q)	CP ↑ to Q(Q_0-Q_7) Invalid Time Delay	0	
13	TdCP(Q)	CP ↑ to Q(Q_0-Q_7) Valid Time Delay (write)		200
14	TdC(Q)	C(C_0-C_2) to Q(Q_0-Q_7) Time Delay—3-state		100
15	TdMR(ER)	$\overline{MR}$ ↓ to ER ↓ Time Delay		200
16	TdCP(ER)	CP ↑ to ER ↑ Valid Time Delay		200
17	TwCPC1	CP Pulse Width (Low) for Correct Functions	450	
18	TwCPCh	CP Pulse Width (High) for Correct Functions	450	
19	TcCPC	CP Cycle Time for Correct Functions	1000	
20	TdC(EP) or TdC(AE)	C(C_0-C_2) to EP or to AE Valid Time Delay		250
21	TdCP(EP) or TdCP(AE)	CP ↓ to EP, to AE or to PM(PM_2-PM_4) Valid Time Delay		400
22	TsP(CP)	P(P_0-P_3) to CP ↓ Setup Time	400	
23	TdP$_0$(EP) or TdP$_0$(AE)	P_0 ↑ to EP or to AE Time Delay		250
24	TsC(CPC) or TsS(CPC)	C(C_0-C_2) or S(S_0-S_1) to CP ↓ Setup Time for Correct Functions	400	
25	TdP(PM)	P_1 or P_2 or P_3 to Corresponding PM Output, Time Delay		250
26	TdCP(EP) or TdCP(AE) or TdCP(PM)	CP ↓ to EP, to AE, or to PM (PM_2, PM_3, PM_4) Invalid Time Delay	0	
27	TdP$_0$(EP) or TdP$_0$(AE)	P_0 ↓ to EP or to AE Invalid Time Delay	0	
28	TwREPh	REP Pulse Width (High)	250	
29	TdREP(Q) or TdREP(LP)	REP ↑ to Q(Q_0-Q_7) or to LP(LP_0-$\overline{LP_3}$) Time Delay		150
30	TdREP(QT) or TdREP(LPT)	REP ↓ to Q(Q_0-Q_7) or to LP(LP_0-LP_3) Time Delay 3-state		100
31	TdP(PM)	P(P_1-P_3) ↓ to PM(PM_2-PM_4) Invalid Time Delay	0	
32	TdC(EP) or TdC(AE) or TdC(PM)	C(C_0-C_2) to EP, or to AE, PM(PM_2-PM_4) Invalid Time Delay	0	

* All timings are preliminary and subject to change.

AC Characteristics (Continued)

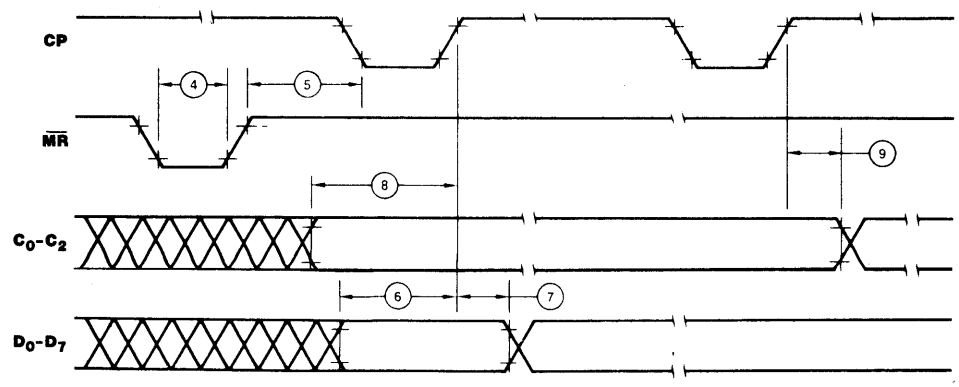
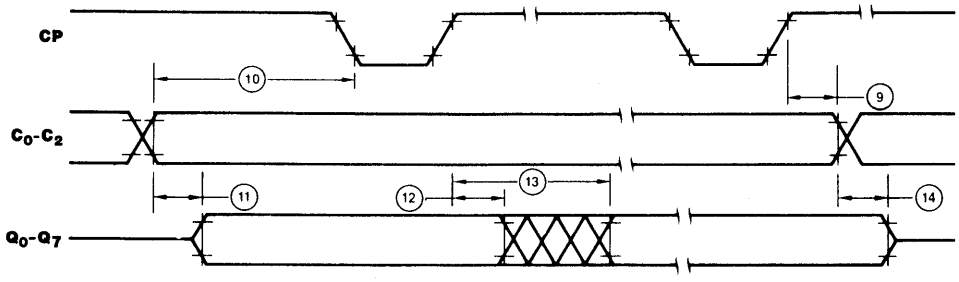
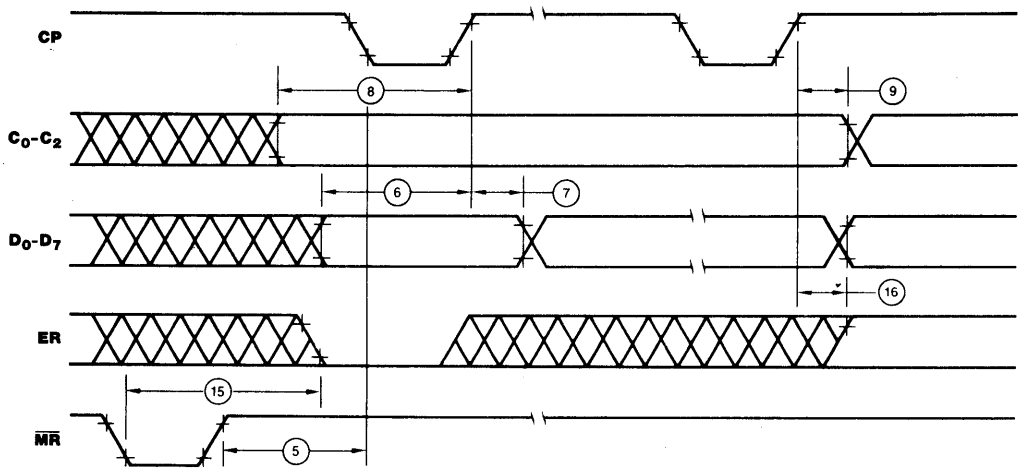


Figure 7. Compute Check Bits or Load Function



Notes: 1. REP input assumed low.
2. Q_0-Q_7 outputs will be high impedance if C_0-C_2 inputs do not specify write check bits function.

Figure 8. Write Check Bits Function



Note: ER output is a function of the contents in the register array flip-flops.

Figure 9. Read Normal or Read High-Speed Function

AC Characteristics (Continued)

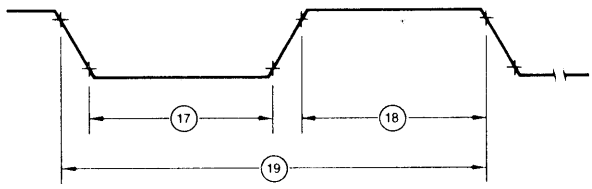
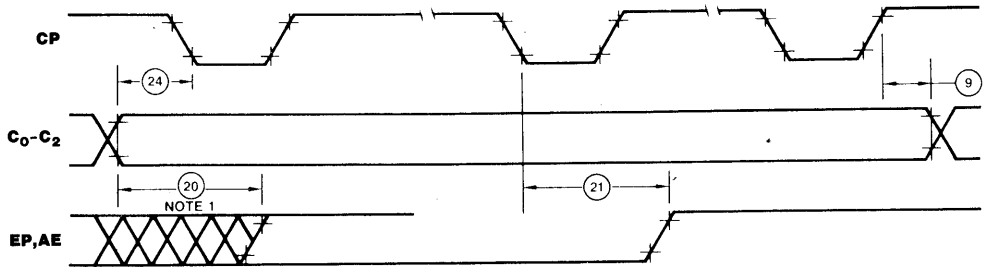
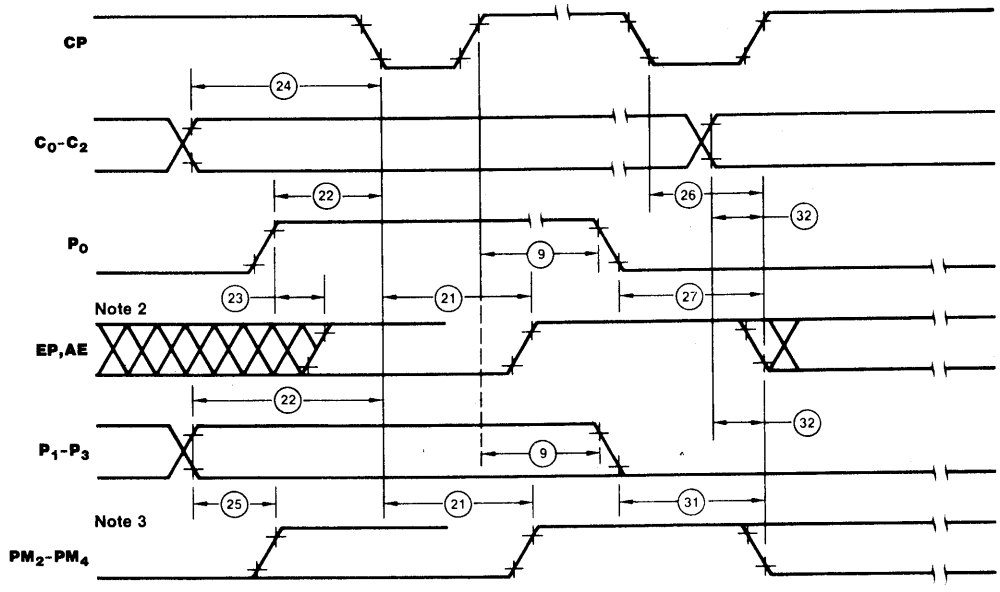


Figure 10. Clock Waveform for Correct Normal or Correct High-Speed Functions



Note 1: Assumes AE or EP output becomes active without any clocking.

Figure 11. Correct Normal Function



Note 2: Assumes EP, AE becomes active without clocking.
Note 3: Assumes corresponding PM output becomes active without clocking.

Figure 12. Correct High-Speed Function

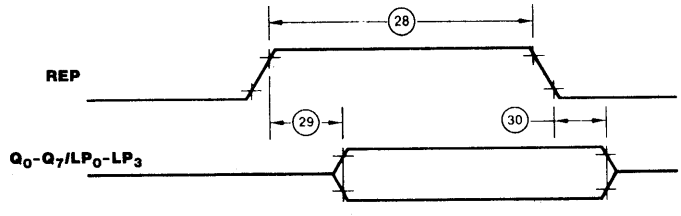


Figure 13. Read Error Pattern Timing

Absolute Maximum Ratings

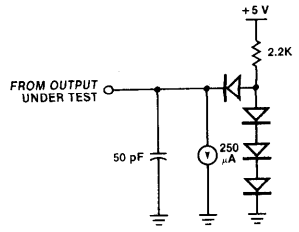
Voltages on all inputs and outputs with respect to GND .....-0.3 V to +7.0 V
Operating Ambient Temperature.....See Ordering Information
Storage Temperature.....-65° to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T_A ≤ +70°C



DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}-0.4	V_{CC}+0.3	V	Driven by external clock generator
V_{CL}	Clock Input Low Voltage	-0.3	0.45	V	Driven by external clock generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 A
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{OL}	Output Leakage		±10	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		300	mA	

Electrical Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IL}	Input Low Voltage	-0.5	+.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OH} = 3.2 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 400 µA
I_{OL}	Output Leakage Current		10	µA	V_{OUT} = 0.4 V
I_{LOH}	Output Leakage Current		10	µA	V_{OUT} = V_{CC}
C_{IN}	Input Capacitance		15	pF	
$C_{I/O}$	I/O Capacitance		25	pF	
I_{LL}	Input Leakage Current		±10	µA	
I_{CC}	V_{CC} Power Supply Current				

NOTE: Typical values apply at T_A = 25°C and V_{CC} = 5.0 V. See table above for operating range.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8065	DS	4.0 MHz	Burst Error Processor (40-pin)	Z8065	PS	4.0 MHz	Burst Error Processor (40-pin)

NOTES: D = Cerdip, P = Plastic, S = 0°C to 70°C

Z8068
Z8000™ Z-DCP Data Ciphering Processor

Zilog

Product Specification

September 1983

Features

- Encrypts and decrypts data using the National Bureau of Standards encryption algorithm.
- Supports three standard ciphering modes: Electronic Code Book, Chain Block and Cipher Feedback.
- Three separate registers for encryption, decryption, and master keys improve system security and throughput by eliminating frequent reloading of keys.
- Three separate programmable ports (master, slave, and key data) provide hardware separation of encrypted data, clear data, and keys.
- Data rates greater than 1M bytes per second can be handled.
- Key parity check.

General Description

The Z8068 Data Ciphering Processor (DCP) is an n-channel, silicon-gate LSI device, which contains the circuitry to encrypt and decrypt data using National Bureau of Standards encryption algorithms. It is designed to be used in a variety of environments, including dedicated controllers, communication concentrators, terminals, and peripheral task processors in general processor systems.

The DCP provides a high throughput rate using Cipher Feedback, Electronic Code Book, or Cipher Block Chain operating modes. The provision of separate ports for key input, clear data, and enciphered data enhances security.

The host system communicates with the DCP using commands entered in the master port or through auxiliary control lines. Once set up, data can flow through the DCP at high speeds because input, output and ciphering activities can be performed concurrently. External DMA control can easily be used to enhance throughput in some system configurations.

The Z8068 DCP is designed to interface directly to Zilog's Z-BUS®. Device signal/pin functions are shown in Figure 1; actual pin number assignments are shown in Figure 2.

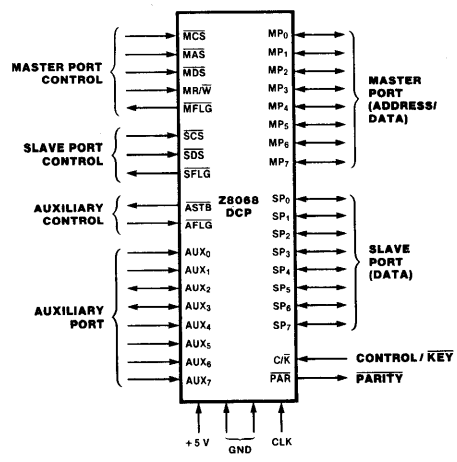


Figure 1. Pin Functions

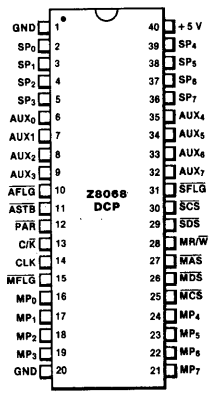


Figure 2. Pin Assignments

Pin Descriptions

$\overline{AFLG}$. *Auxiliary Port Flag* (output, active Low). This output signal indicates that the DCP is expecting key data to be entered on pins AUX_0-AUX_7. This can occur only when C/$\overline{K}$ is Low and a "Load Key Through AUX Port" command has been entered. $\overline{AFLG}$ remains active (Low) during the input of all eight bytes and will go inactive with the leading edge of the eighth strobe ($\overline{ASTB}$).

$\overline{ASTB}$. *Auxiliary Port Strobe* (input, active Low). In Multiplexed Control mode (C/$\overline{K}$ Low), the rising (trailing) edge of $\overline{ASTB}$ strobes the key data on pins AUX_0-AUX_7 into the appropriate internal key register. This input is ignored unless $\overline{AFLG}$ and C/$\overline{K}$ are both Low. One byte of key data is entered on each $\overline{ASTB}$ with the most significant byte entered first.

AUX_0-AUX_7. *Auxiliary Port Bus* (bidirectional, active High). When the DCP is operated in Multiplexed Control mode (C/$\overline{K}$ Low), these eight lines form a key-byte input port, which can be used to enter the master and session keys. This port is the only path available for entering the master key. (Session keys can also be entered via the master port.) AUX_0 is the low-order bit and is considered to be the parity bit in key bytes. The most significant byte is entered first.

When the DCP is operated in Direct Control mode (C/$\overline{K}$ High), the auxiliary port's key-entry function is disabled and five of the eight lines become direct control/status lines for interfacing to high-speed microprogrammed controllers. In this case, AUX_0, AUX_1 and AUX_4 have no function, and the other pins are defined as follows:

AUX_2-$\overline{BSY}$. *Busy* (output, active Low). This status output gives a hardware indication that the ciphering algorithm is in operation. AUX_2-$\overline{BSY}$ is driven by the BSY bit in the Status register such that when the BSY bit is 1 (active), AUX_2-$\overline{BSY}$ is Low.

AUX_3-$\overline{CP}$. *Command Pending* (output, active Low). This status output gives a hardware indication that the DCP is ready to accept the input of key bytes following a Low-to-High transition on AUX_7-K/$\overline{D}$. AUX_3-$\overline{CP}$ is driven by the CP bit in the Status register such that when the CP bit is 1 (active), AUX_3-$\overline{CP}$ is Low.

AUX_5-S/$\overline{S}$. *Start/Stop* (input, Low = Stop). When this pin goes Low (Stop), the DCP follows the normal Stop command sequence. When this pin goes High, a sequence equivalent to a Start Encryption or Start Decryption command is followed. When AUX_5-S/$\overline{S}$ goes High, the level on AUX_6-E/$\overline{D}$ selects either the start encryption or start decryption operation.

AUX_6-E/$\overline{D}$. *Encrypt/Decrypt* (input, Low = Decrypt). When AUX_5-S/$\overline{S}$ goes High, it initiates a normal data ciphering operation whose input specifies whether the ciphering algorithm is to encrypt (E/$\overline{D}$ High) or decrypt (E/$\overline{D}$ Low).

When AUX_7-K/$\overline{D}$ goes High, initiating the entry of key bytes, the level on AUX_6-E/$\overline{D}$ specifies whether the bytes are to be written into the E Key register (E/$\overline{D}$ High) or the D key Register (E/$\overline{D}$ Low).

The AUX_6-E/$\overline{D}$ input is not latched internally and must be held constant whenever one or more of AUX_5-S/$\overline{S}$, AUX_7-K/$\overline{D}$, AUX_2-$\overline{BSY}$, or AUX_3-$\overline{CP}$ are active. Failure to maintain the proper level on AUX_6-E/$\overline{D}$ during loading or ciphering operations results in scrambled data in the internal registers.

AUX_7-K/$\overline{D}$. *Key/Data* (input, Low = Data). When this signal goes High, the DCP initiates a key-data input sequence as if a Load Clear E or D Key Through Master Port command had been entered. The level on AUX_6-E/$\overline{D}$ determines whether the subsequently entered clear-key bytes are written into the E key register (E/D High) or the D key register (E/$\overline{D}$ Low)

AUX_7-K/$\overline{D}$ and AUX_5-S/$\overline{S}$ are mutually exclusive control lines; when one goes active (High), the other must remain inactive (Low) until the first returns to an inactive state. In addition, both lines must be inactive (Low) whenever a transition occurs on C/$\overline{K}$ (entering or exiting Direct Control mode).

C/$\overline{K}$. *Control/Key Mode Control.* (input, Low = Key). This input determines the operating characteristics of the DCP. A Low input on C/$\overline{K}$ puts the DCP into the Multiplexed Control mode, enabling programmed access to internal registers through the master port and enabling input of keys through the master or auxiliary port. A High input on C/$\overline{K}$ specifies operation in Direct Control mode. In this mode, several of the auxiliary port pins become direct control status signals which can be driven/sensed by high-speed controller logic, and access to internal registers through the master port is limited to the Input or Output register.

CLK. *Clock* (input, TTL compatible). An external timing source is input via the CLK pin. The Data Strobe signals ($\overline{MDS}$, $\overline{SDS}$) must change synchronously with this clock input, as must Master Port Address Strobe ($\overline{MAS}$) in Multiplexed Control mode (C/$\overline{K}$ Low), and also AUX_7-K/$\overline{D}$ and AUX_5-S/$\overline{S}$ in Direct Control mode (C/$\overline{K}$ High). In addition, the Auxiliary, Master and Slave Port Flag outputs ($\overline{AFLG}$, $\overline{MFLG}$, and $\overline{SFLG}$) change synchronously with the clock. When using the DCP with the Z8000 CPU in Multiplexed Control mode, the clock input must agree in frequency and phase with the processor clock; however, the DCP does not require the high voltage levels of the processor clock.

Pin Descriptions (Continued)

$\overline{\text{MAS}}$. *Master Port Address Strobe* (input, active Low). In Multiplexed Control mode (C/$\overline{\text{K}}$ Low), an active (Low) signal on this pin indicates the presence of valid address and chip select information at the master port. This information is latched internally on the rising edge of Master Port Address Strobe ($\overline{\text{MAS}}$). When C/$\overline{\text{K}}$ is High (Direct Control mode), $\overline{\text{MAS}}$ can be High or Low without affecting DCP operation, except that, regardless of the state of C/$\overline{\text{K}}$, if both Master Port Address Strobe ($\overline{\text{MAS}}$) and Data Strobe ($\overline{\text{MDS}}$) are Low simultaneously, the DCP Mode register will be reset to ECB mode. The master port is assigned to clear data, the slave port is assigned to enable data, and all flags remain inactive.

$\overline{\text{MCS}}$. *Master Port Chip Select* (input, active High). This signal is used to select the master port. In Multiplexed Control mode (C/$\overline{\text{K}}$ Low), the level on $\overline{\text{MCS}}$ is latched internally on the rising edge of Master Port Address Strobe ($\overline{\text{MAS}}$). This latched level is retained as long as $\overline{\text{MAS}}$ is High; when $\overline{\text{MAS}}$ is Low, the latch becomes invisible and the internal signal follows the $\overline{\text{MCS}}$ input. In Direct Control mode (C/$\overline{\text{K}}$ High), no latching of Master Port Chip Select occurs; the level on $\overline{\text{MCS}}$ is passed directly to the internal select circuitry, regardless of the state of Address Strobe ($\overline{\text{MAS}}$).

$\overline{\text{MDS}}$. *Master Port Data Strobe* (input, active Low). When $\overline{\text{MDS}}$ is active and Master Port Chip Select ($\overline{\text{MCS}}$) is valid, it indicates that valid data is present on MP_0-MP_7 during output. $\overline{\text{MDS}}$ and Master Port Address Strobe ($\overline{\text{MAS}}$) are normally mutually exclusive; if both go Low simultaneously, the DCP is reset to ECB mode and all flags remain inactive.

$\overline{\text{MFLG}}$. *Master Port Flag* (output, active Low). This flag is used to indicate the need for a data transfer into or out of the master port during normal ciphering operation. Depending upon the control bits written to the Mode register, the master port is associated with either the Input register or the Output register.

If data is to be transferred through the master port to the Input register, the $\overline{\text{MFLG}}$ reflects the contents of the Input register; after any start command is entered, $\overline{\text{MFLG}}$ goes active (Low) whenever the Input register is not full. $\overline{\text{MFLG}}$ is forced High by any command other than a start. Conversely, if the master port is associated with the Output register, $\overline{\text{MFLG}}$ reflects the contents of the Output register (except in single-port configuration). $\overline{\text{MFLG}}$ goes active (Low) whenever the Output register is not empty. In single-port configuration, $\overline{\text{MFLG}}$ reflects the contents of the Input register, while the Slave Port Flag ($\overline{\text{SFLG}}$) is associated with the Output register.

MP_0-MP_7. *Master Port Bus* (input/output, active High). These eight bidirectional lines are used to specify internal register addresses in Multiplexed Control mode (see C/$\overline{\text{K}}$) and to input and output data. The master port provides software access to the Status, Command and Mode registers as well as the Input and Output registers. The 3-state master port outputs are enabled only when the master port is selected by Master Port Chip Select ($\overline{\text{MCS}}$) being Low, with Master Port Read/Write (MR/$\overline{\text{W}}$) High, and strobed by a Low on the Master Port Data Strobe ($\overline{\text{MDS}}$). MP_0 is the low-order bit. Data and key information is entered into this port with most significant byte input first.

MR/$\overline{\text{W}}$. *Master Port Read/Write* (input, Low = Write). This signal indicates to the DCP whether the current master port operation is a read (MR/$\overline{\text{W}}$ is High) or a write (MR/$\overline{\text{W}}$ is Low), thereby indicating whether data is to be transferred from or to an internal register. MR/$\overline{\text{W}}$ is not latched internally and must be held stable while Master Port Data Strobe ($\overline{\text{MDS}}$) is Low.

$\overline{\text{PAR}}$. *Parity* (output, active Low). The DCP checks all key bytes for correct (odd) parity as they are entered through either the master port (Multiplexed or Direct Control mode) or the auxiliary port (Multiplexed Control mode only). If any key byte contains even parity, the PAR bit in the Status register is set to 1 and PAR goes Low. The least significant bit of key bytes is the parity.

$\overline{\text{SCS}}$. *Slave Port Chip Select* (input, active Low). This signal is logically combined with Slave Port Data Strobe ($\overline{\text{SDS}}$) to facilitate slave port data transfers in a bus environment. $\overline{\text{SCS}}$ is not latched internally and can be permanently tied to Low without impairing slave port operation.

$\overline{\text{SDS}}$. *Slave Port Data Strobe* (input, active Low). When both $\overline{\text{SDS}}$ and $\overline{\text{SCS}}$ are Low, it indicates to the DCP either that valid data is on the SP_0-SP_7 lines for an input operation, or that data is to be driven onto the SP_0-SP_7 lines for output. The direction of data flow is determined by the control bits in the Mode register.

$\overline{\text{SFLG}}$. *Slave Port Flag* (output, active Low). This output indicates the status of either the Input register or the Output register, depending on the control bits in the Mode register. In single-port configuration, $\overline{\text{SFLG}}$ goes active during normal processing whenever the Output register is not empty. In dual-port configuration, $\overline{\text{SFLG}}$ reflects the content of whichever register is associated with the slave port. If the input register is assigned to the slave port, $\overline{\text{SFLG}}$ goes active whenever the Input register is not full, once any of the start commands has been entered; $\overline{\text{SFLG}}$ is forced

Pin Descriptions (Continued)

inactive if any other command is entered. If the slave port is assigned to the Output register, $\overline{SFLG}$ goes active whenever the Output register is not empty. In this case, $\overline{SFLG}$ goes inactive if any command is aborted.

SP_0-SP_7. *Slave Port Bus* (bidirectional). The slave port provides a second data input/output interface to the DCP, allowing overlapped input, output, and ciphering operations. The 3-state slave port outputs are driven only when Slave Port Chip Select ($\overline{SCS}$) and Slave Port Data Strobe ($\overline{SDS}$) are both Low, $\overline{SFLG}$ is 0, and the internal port control configuration allows output to the slave port. SP_0 is the low order bit. The most significant byte of data blocks is entered or retrieved through this port first.

Functional Description

The overall design of the DCP, as shown in Figure 3, is optimized to achieve high data throughput. Data bytes can be transferred through both the master and slave ports, and key bytes can be written through both the auxiliary and master ports. Three 8-bit buses (input, output and C bus) carry data and key bytes between the ports and the internal registers. Three 56-bit, write-only key registers are provided for the Master (M) Key, the Encryption (E) Key and the Decryption (D) Key. Parity checking is provided on incoming key bytes. Two 64-bit registers are provided for initializing vectors (IVE and IVD) that are required for chained (feedback) ciphering modes. Three 8-bit registers (Mode, Command and Status) are accessible through the master port.

Algorithm Processing. The algorithm processing unit of the DCP (Figure 3) is designed to encrypt and decrypt data according to the National Bureau of Standards' Data Encryption Standard (DES), as specified in Federal Information Processing Standards Publication 46. The DES specifies a method for encrypting 64-bit blocks of clear data ("plain text") into corresponding 64-bit blocks of "cipher text."

The DCP offers three ciphering methods, selected by the cipher type field of the Mode register: Electronic Code Book (ECB), Cipher Block Chain (CBC) and Cipher Feedback (CFB). These methods are implemented in accordance with Federal Information Processing Standards, Publication 46.

Electronic Code Book (ECB) is a straightforward implementation of the DES: 64 bits of clear data in, 64 bits of cipher text out, with no cryptographic dependence between blocks.

Cipher Block Chain (CBC) also operates on blocks of 64 bits, but it includes a feedback step which chains consecutive blocks so that repetitive data in the plain text (such as ASCII blanks) does not yield repetitive cipher text. CBC also provides an error extension characteristic which protects against fraudulent data insertions and deletions.

Cipher Feedback (CFB) is an additive stream cipher method in which the DES algorithm generates a pseudorandom binary stream, which is then exclusive-ORed with the clear data to form the cipher text. The cipher text is then fed back to form a portion of the next DES input block. The DCP implements 8-bit cipher feedback, with data input, output,

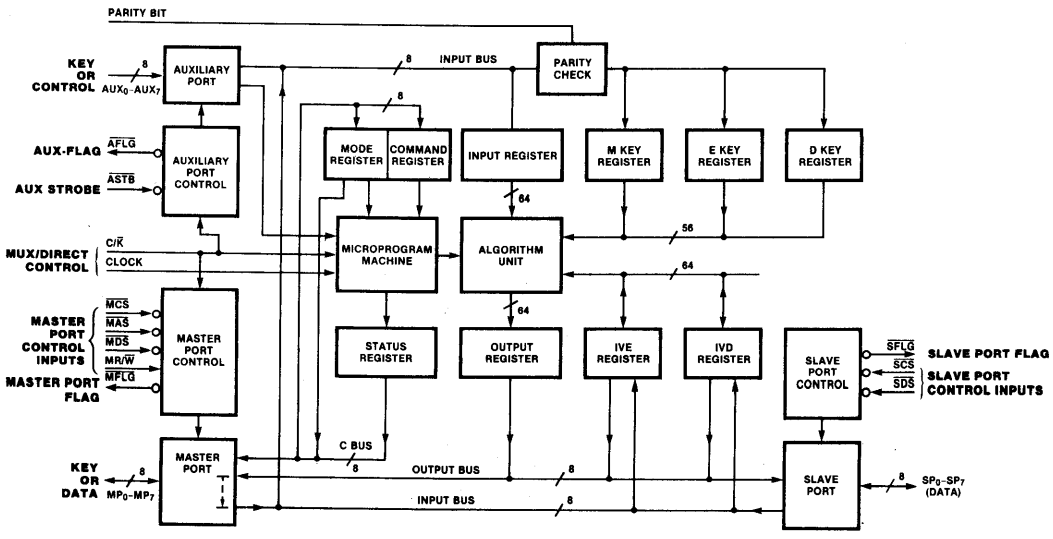


Figure 3. Z8068 Block Diagram

Functional Description (Continued)

and feedback paths of one byte wide. This method is useful for low speed, character-at-a-time, serial communications.

Multiple Key Registers. The DCP provides the necessary registers to implement a multiple-key or master-key system. In such an arrangement, a single master key, stored in the DCP M key register, is used to encrypt session keys for transmission to remote DES equipment and to decrypt session keys received from such equipment. The M Key register may be loaded (with plain text) only through the auxiliary port, using the Load Clear Master Key command. In addition to the M Key register, the DCP contains two session key registers: the E key register, used to encrypt clear text, and the D key register, used to decrypt cipher text. All three registers are loaded by writing commands such as Load Clear E Key, through master port, into the Command register, and then writing the eight bytes of key data to the port when the Command Pending bit in the Status register is 1.

Operating Modes: Multiplexed Control vs. Direct Control. The DCP can be operated in either of two basic interfacing modes, determined by the logic level on the C/$\overline{K}$ input pin. In Multiplexed Control mode (C/$\overline{K}$ Low), the DCP is configured internally to allow a master CPU to address five of the internal control/status/data registers directly, thereby controlling the device via mode and command values written to these registers. Also, in this mode, the auxiliary port is enabled for key-byte input.

If the logic level on C/$\overline{K}$ is brought High, the DCP enters Direct Control mode, and the auxiliary port pins are converted into direct hardware status or control signals capable of instructing the DCP to perform a functionally complete subset of its cipher processing at very high throughputs. This operating mode is particularly well suited for ciphering data for high-speed peripheral devices such as magnetic disk or tape.

Data Flow. Bits M_2 and M_3 of the Mode register control the flow of data into and out of the DCP through the master and slave ports. Three basic configurations are provided: one single-port and two dual-port.

Single-Port Configuration. The simplest configuration occurs when the Mode register configuration bits are set to master port only (Figure 4). In this operating configuration, the encrypt/decrypt bit (M_4) controls the processing of data. Data to be encrypted or decrypted is written to the master port Input register address. To facilitate monitoring of the Input register status, the $\overline{MFLG}$ signal goes Low when the Input register is not full. Data is read by the master CPU through the master port Output register address. Pin $\overline{SFLG}$ goes Low when the Output register is not empty. $\overline{MFLG}$ is then redefined as a master input flag and $\overline{SFLG}$ is redefined as a master output flag.

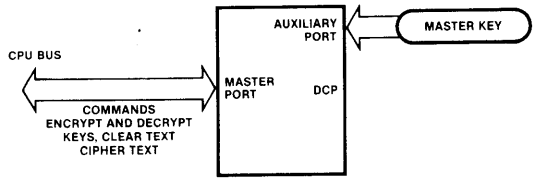


Figure 4. Single-Port Configuration, Multiplexed Control

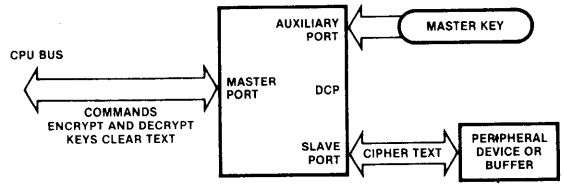


Figure 5a. Dual-Port Configuration, Multiplexed Control

Dual Port, Master Port Clear Configuration. In the dual-port configurations, both the master and slave ports are used for data entry and removal (Figures 5a and 5b). In the master port clear configuration, clear text for encryption can be entered only through the master port, and clear text resulting from decryption can be read only through the master port. Cipher text can be handled only through the slave port. The actual direction of data flow is controlled either by the encrypt/decrypt bit (M_4) in the Mode register or by the Start Encryption or Start Decryption commands. If encryption is specified, clear data will flow through the master port to the Input register, and cipher data will be available at the slave port when it is ready to be read from the Output register. For decryption, the process is reversed, with cipher data written to the Input register

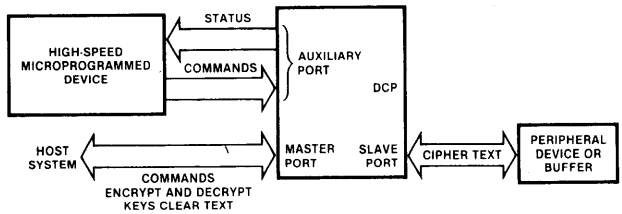


Figure 5b. Dual-Port Configuration, Direct Control

Functional Description (Continued)

through the master port. Slave port and clear text read from the Master port.

In both dual-port configurations, the Master Port Flag ($\overline{MFLG}$) and the Slave Port Flag ($\overline{SFLG}$) are used to indicate the status of the data register associated with the master port and slave port, respectively. For example, during encryption in the master port clear configuration, $\overline{MFLG}$ goes Low (active) when the Input register is not full; $\overline{SFLG}$ goes Low (active) when the Output register is not empty. If cyphering operation changes direction, $\overline{MFLG}$ and $\overline{SFLG}$ switch their register association (see Table 1).

Mode Register Bits				
Encrypt/ Decrypt Bit M4	Port Configuration Bit M3	Bit M2	Input Register Flag	Output Register Flag
0	0	0	$\overline{MFLG}$	$\overline{SFLG}$
0	0	1	$\overline{SFLG}$	$\overline{MFLG}$
0	1	0	$\overline{MFLG}$	$\overline{SFLG}$
1	0	0	$\overline{SFLG}$	$\overline{MFLG}$
1	0	1	$\overline{MFLG}$	$\overline{SFLG}$
1	1	0	$\overline{MFLG}$	$\overline{SFLG}$

Table 1. Association of Master Port Flag (MFLG) and Slave Port Flag (SFLG) with Input and Output Registers

Dual Port, Slave Port Clear Configuration. This configuration is identical to the previously described dual-port, master port clear configuration except that the direction of ciphering is reversed. That is, all data flowing in or out of the master port is cipher text, and all data at the slave port is clear text.

Master Port Read/Write Timing. The master port of the DCP is designed to operate directly with a multiplexed address/data bus such as the Zilog Z-BUS. Several features of the master port logic are:

- The level on Master Port Chip Select ($\overline{MCS}$) is latched internally on the rising (trailing) edge of Master Port Address Strobe ($\overline{MAS}$). This action relieves external address decode circuitry of the responsibility for latching chip select at address time.
- The levels on MP_1 and MP_2 are also latched internally on the rising edge of $\overline{MAS}$ and are subsequently decoded to enable reading and writing of the DCP's internal registers (Mode, Command, Status, Input and Output). This action also eliminates the need for external address latching and decoding.
- Data transfers through the master port are controlled by the levels and transitions on Master Port Data Strobe ($\overline{MDS}$) and Master Port Read/Write (MR/$\overline{W}$). The former controls the timing and the latter controls the transfer direction. Data transfers disturb neither the chip-select nor address latches, so once the DCP and a particular register have been selected, any number of reads or writes of that register can be accomplished without intervening address cycles. This feature greatly speeds up the loading of keys and data, given the necessary transfer control external to the DCP.

Loading Keys and Initializing Vector (IV) Registers. Because the key and Initializing Vector (IV) registers are not directly addressable through any of the DCP's ports, keys and vector data must be loaded (and in the case of vectors, read) via "command data sequences." Most of the commands recognized by the DCP are of this type. A load or read command is written to the Command register through the master port. The command processor responds by asserting the Command Pending output. The user then either writes eight bytes of key or vector data through the master or auxiliary port, as appropriate to the specific command, or reads eight bytes of vector data from the master port.

In Direct Control mode, only the E Key and D Key registers can be loaded; the M Key and IV registers are inaccessible. Loading the E and D Key registers is accomplished by placing the proper state on the AUX_6-E/$\overline{D}$ input (High for E Key, Low for D Key) and then raising the AUX_7-K/$\overline{D}$ input—indicating that key loading is required. The command processor attaches the proper key register to the master port and asserts the AUX_3-$\overline{CP}$ (Command Pending) signal (active Low). The eight key bytes can then be written to the master port. In the Multiplexed Control mode, all key and vector registers can be written to and all but the Master (M) Key register can be loaded with encrypted, as well as clear, data. If the operation is a Load Encrypt command, the subsequent data written to the master or auxiliary port (as appropriate) is routed first to the Input register and decrypted before it is written into the specified key or Initializing Vector register.

Parity Checking of Keys. Key bytes contain seven bits of key information and one parity bit. By DES designation, the low-order bit is the parity bit. The parity-check circuit is enabled whenever a byte is written to one of three key registers. The output of the parity-check circuit is connected to $\overline{PAR}$ and the state of this signal is reflected in Status register bit PAR (S_3). Status register bit PAR goes to 1 whenever a byte with even parity (an even number of 1s) is detected. In addition to the PAR bit, the Status register has a Latched Parity bit (LPAR, S_4) that is set to 1 whenever the Status register PAR bit goes to 1. Once set, the LPAR bit is not cleared until a reset occurs or a new Load Key command is issued.

Functional Description (Continued)

When an encrypted key is entered, the parity-check logic operates only after the decrypted key is available. The encrypted data is not checked for parity. The $\overline{PAR}$ signal reflects the state of the decrypted bytes on a byte-to-byte basis as they are clocked through the parity-check logic on their way to the key register. Thus, the time during which $\overline{PAR}$ indicates the status of a byte of decrypted key data may be as short as four clock cycles. The LPAR bit in the Status register indicates if any erroneous bytes of key data were entered.

Programming

Initialization. The DCP can be reset in several ways:

- By the "Software Reset" command.
- By a hardware reset, which occurs whenever both $\overline{MAS}$ and $\overline{MDS}$ go Low simultaneously.
- By writing to the Mode register.
- By aborting any command.

These sequences initiate the same internal operations, except that loading the Mode register or aborting any command does not subsequently reset the Mode register. Once a reset process starts, the DCP is unable to respond to further commands for approximately five clock cycles. If a power-up hardware reset is used, the leading edge of the reset signal should not occur until approximately 1 ms after V_{CC} has reached normal operating voltage. This delay time is needed for internal signals to stabilize.

Registers. The registers in the DCP that can be addressed directly through the master port are shown with their addresses in Table 2. A brief description of these registers and those not directly accessible follows.

C/$\overline{K}$	MP2	MP1	MR/$\overline{W}$	$\overline{MCS}$	Register Addressed
0	X	0	0	0	Input Register
0	X	0	1	0	Output Register
0	0	1	0	0	Command Register
0	0	1	1	0	Status Register
0	1	1	X	0	Mode Register
X	X	X	X	1	No Register Accessed
1	X	X	0	0	Input Register
1	X	X	1	0	Output Register

Table 2. Master Port Register Addresses

Hex Code	Command
90	Load Clear M Key Through Auxiliary Port
91	Load Clear E Key Through Auxiliary Port
92	Load Clear D Key Through Auxiliary Port
11	Load Clear E Key Through Master Port
12	Load Clear D Key Through Master Port
B1	Load Encrypted E Key Through Auxiliary Port
B2	Load Encrypted D Key Through Auxiliary Port
31	Load Encrypted E Key Through Master Port
32	Load Encrypted D Key Through Master Port
85	Load Clear IVE Through Master Port
84	Load Clear IVD Through Master Port
A5	Load Encrypted IVE Through Master Port
A4	Load Encrypted IVD Through Master Port
8D	Read Clear IVE Through Master Port
8C	Read Clear IVD Through Master Port
A9	Read Encrypted IVE Through Master Port
A8	Read Encrypted IVD Through Master Port
39	Encrypt With Master Key
41	Start Encryption
40	Start Decryption
C0	Start
E0	Stop
00	Software Reset

Table 3. Command Codes in Multiplexed Control Mode

Command Register. Data written to the 8-bit, write-only Command register through the master port is interpreted as an instruction. A detailed description of each command is given in the Commands section; the commands and their hexadecimal representations are summarized in Table 3. A subset of these commands can be entered implicitly in Direct Control mode (C/$\overline{K}$ High)—even though the Command register cannot be addressed in that mode—by transitions on auxiliary lines AUX_5-S/$\overline{S}$, AUX_6-E/$\overline{D}$, and AUX_7-K/$\overline{D}$. These implicit commands are summarized in Table 4.

	Pins			
C/$\overline{K}$	AUX_7-K/$\overline{D}$	AUX_6-E/$\overline{D}$	AUX_5-S/$\overline{S}$	Command Initiated
H	L	L	↑	Start Decryption
H	L	H	↑	Start Encryption
H	L	X	↓	Stop
H	↑	L	L	Load D Key Clear through master port
H	↑	H	L	Load E Key Clear through master port
H	↓	X	L	End Load Key command
H	H	X	H	Not allowed
L	Data	Data	Data	AUX pins become Key-Byte inputs

Table 4. Implicit Command Sequences in Direct Control Mode

Programming (Continued)

Status Register. The bit assignments in the read-only Status register are shown in Figure 6. The PAR, AFLG, SFLG and MFLG bits indicate the status of the corresponding output pins, as do the busy and command pending bits when the DCP is in a Direct Control mode (C/$\overline{K}$ High). In each case, the output signal will be active Low when the corresponding status bit is a 1. The parity bit indicates the parity of the most recently entered key byte. The LPAR bit indicates whether any key byte with even parity has been encountered since the last Reset or Load Key command.

The Busy bit is 1 whenever the ciphering algorithm unit is actively encrypting or decrypting data, either as a response to a command such as Load Encrypted Key (in which case the Command Pending bit is 1) or in the ciphering of regular text (indicated by the Start/Stop bit being 1). If the ciphered data cannot be transferred to the Output register because that register still contains output from a previous ciphering cycle, the Busy bit remains 1 even after the ciphering is complete. Busy is 0 at all other times, even when ciphering is not possible because data has not been written to the Input register.

The Command Pending bit is set to 1 by any command whose execution requires the transfer of data to or from a nonaddressable internal register, such as when writing key bytes to the E key register or reading bytes from the IVE register. Thus, the Command Pending bit is set following all commands except the three start commands, the Stop command and the Software Reset command. The Command Pending bit returns to 0 after all eight bytes have been transferred following Load Clear, Read Clear, or Read Encrypted commands; and after data has been transferred, decrypted, and loaded into the desired register following Load Encrypt commands.

The Start/Stop bit is set to 1 when one of the start commands is entered and it is reset to 0 *whenever a reset occurs or when a new command* other than a Start is entered.

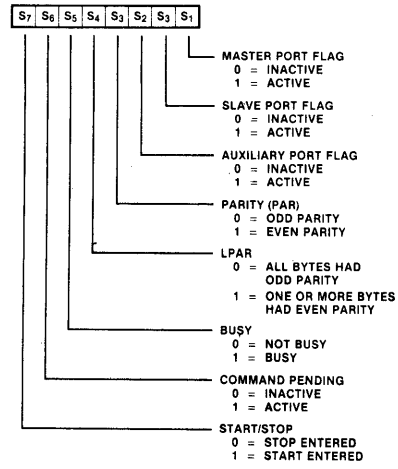


Figure 6. Status Register Bit Assignments

Mode Register. Bit assignments in this 5-bit read/write register are shown in Figure 7. The cipher type bits (M_1 and M_0) indicate to the DCP which ciphering algorithm is to be used. On reset, the Cipher Type mode defaults to Electronic Code Book mode.

Configuration bits (M_3 and M_2) indicate which data ports are to be associated with the Input and Output registers and flags. When these bits are set to the single-port, master-only configuration (M_3 M_2 = 10), the slave port is disabled and no manipulation of Slave Port Chip Select ($\overline{SCS}$) or Slave Data Strobe ($\overline{SDS}$) can result in data movement through the slave port; all data transfers are accomplished through the master port, as previously described in the Functional Description. Both $\overline{MFLG}$ and $\overline{SFLG}$ are used in this configuration; $\overline{MFLG}$ gives the status of the Input register and $\overline{SFLG}$ gives the status of the Output register.

When the configuration bits are set to one of the dual-port configurations (M_3 M_2 = 00 or 01), both the master and slave ports are available for input and output. When M_3, M_2 = 01 (the default configuration), the master port handles clear data while the slave port handles encrypted data. Configuration M_3, M_2 = 00 reverses this assignment. Actual data direction at any particular moment is controlled by the Encrypt/Decrypt bit.

The Encrypt/Decrypt bit (M_4) instructs the DCP algorithm processor to encrypt or decrypt the data from the Input register using the ciphering method specified by the Cipher Type bits. The Encrypt/Decrypt bit also controls data flow within the DCP. For example, when the configuration bits are 0,1 (dual-port, master clear, slave encrypted) and the Encrypt/Decrypt bit is 1 (encrypt), clear data will flow into the DCP through the master port and encrypted data will flow out through the slave port. When the Encrypt/Decrypt bit is set to 0 (decrypt), data flow is reversed.

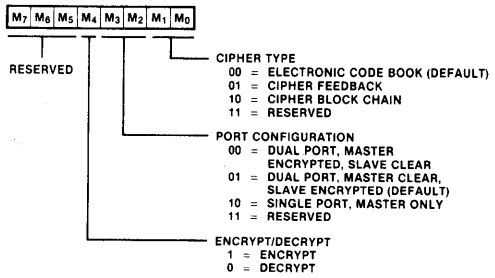


Figure 7. Mode Register Bit Assignments

Programming (Continued)

Input Register. The 64-bit, write-only Input register is organized to appear to the user as eight bytes of pushdown storage. A status circuit monitors the number of bytes that have been stored. The register is considered empty when the data stored in it has been or is being processed; it is considered full when one byte of data has been entered in Cipher Feedback mode or when eight bytes of data have been entered in Electronic Code Book or Cipher Block Chain mode. If the user attempts to write data into the Input register when it is full, the Input register disregards the attempt; no data in the register is destroyed.

Output Register. The 64-bit, read-only Output register is organized to appear to the user as eight bytes of pop-up storage. A status circuit detects the number of bytes stored in the Output register. The register is considered empty when all the data stored in it has been read by the master CPU and is considered full if it still contains one or more bytes of output data. If a user attempts to read data from the Output register when it is empty, the buffers driving the output bus remain in a 3-state condition.

M, E, D Key Registers. The following multibyte key registers cannot be addressed directly, but are loaded in response to commands written to the Command register.

There are three 64-bit, write-only key registers in the DCP: the Master (M) Key register, the Encrypt (E) key register, and the Decrypt (D) key register. The Master key register can be loaded only with clear data through the auxiliary port. The Encrypt and Decrypt Key registers can be loaded in any of four ways: (1) as clear data through the auxiliary port, (2) as clear data through the master port, (3) as encrypted data through the auxiliary port, or (4) as encrypted data through the master port. In the last two cases, the encrypted data is first routed to the Input register, decrypted using the M Key, and finally written to the target key register from the Output register.

Initializing Vector Registers (IVE and IVD). Two 64-bit registers are provided to store feedback values for cipher feedback and chained block ciphering methods. One initializing vector register (IVE) is used during encryption, the other (IVD) is used during decryption. Both registers can be loaded with either clear or encrypted data through the master port (in the latter case, the data is decrypted before being loaded into the IV register), and both may be read out either clear or encrypted through the master port.

Commands

All operations of the DCP result from command inputs, which are entered in Multiplexed Control mode by writing a command byte to the Command register. Command inputs are entered in Direct Control mode by raising and lowering the logic levels on the AUX_7-K/$\overline{D}$, AUX_6-E/$\overline{D}$, and AUX_5-S/$\overline{S}$ pins. Table 3 shows all commands that can be given in Multiplexed Control mode. Table 4 shows a subset of the implicit commands that can be executed in the Direct Control mode.

Load Clear M Key Through Auxiliary Port (90H).
Load Clear E Key Through Auxiliary Port (91H).
Load Clear D Key Through Auxiliary Port (92H).

These commands may be used only for multiplexed operations; they override the data flow specifications set in the Mode register and cause the Master (M) Key, Encrypt (E) Key, or Decrypt (D) Key register to be loaded with eight bytes written to the auxiliary port. After the Load command is written to the Command register, the Auxiliary Port Flag ($\overline{AFLG}$) goes active (Low) and the corresponding bit in the Status register (S_2) becomes 1, indicating that the device is able to accept key bytes at the auxiliary port pins. Additionally, the Command Pending bit (S_6) becomes 1 during the entire loading process.

Each byte is written to its respective key register by placing an active Low signal on the Auxiliary Port Strobe ($\overline{ASTB}$) once data has been set up on the auxiliary port pins. The actual write process occurs on the rising (trailing) edge of $\overline{ASTB}$. (See Switching Characteristics section for exact setup, strobe width, and hold times.)

The Auxiliary Port Flag ($\overline{AFLG}$) goes inactive immediately after the eighth strobe goes active (Low). However, the Command Pending bit (S_6) remains 1 for several more clock cycles, until the key loading process is completed. All key bytes are checked for correct (odd) parity as they are entered.

Load Clear E Key Through Master Port (11H).
Load Clear D Key Through Master Port (12H).

These commands are available in both Multiplexed Control and Direct Control modes. They override the data flow specifications set in the Mode register and attach the master port inputs to the Encrypt (E) Key or Decrypt (D) Key register, as appropriate, until eight key bytes have been written. In Multiplexed Control mode, the command is initiated by writing the Load command to the Command register. In Direct Control mode, the command is initiated by raising the AUX_7-K/$\overline{D}$ control input while the AUX_5-S/$\overline{S}$

Commands (Continued)

input is Low. In this latter case, the level on AUX_6-E/$\overline{D}$ determines which key register is written (High = E register).

Once the command has been recognized, the Command Pending bit (S_6 in the Status register) becomes 1. In Direct Control mode, AUX_3-$\overline{CP}$ goes active (Low), indicating that key entry may proceed. The host system then writes exactly eight bytes to the master port (at the Input register address in Multiplexed Control mode). When the key register has been loaded, the Command Pending bit returns to 0. In Direct Control mode, the AUX_3-$\overline{CP}$ output goes inactive, indicating that the DCP can accept the next command.

Load Encrypted E Key Through Auxiliary Port (B1H).
Load Encrypted D Key Through Auxiliary Port (B2H).

These commands are used in Multiplexed Control mode only. Their execution is similar to that of the Load Clear E (D) Key Through Auxiliary Port command, except that key bytes are first decrypted using the electronic code book algorithm and the Master (M) Key register. The key bytes are then loaded into the appropriate key register, after having passed through the parity-check logic.

The Command Pending bit (S_6) is 1 during the entire decrypt-and-load operation. In addition, the Busy bit (S_5) is 1 during the actual decryption process.

Load Encrypted E Key Through Master Port (31H).
Load Encrypted D Key Through Master Port (32H).

These commands are used in Multiplexed Control mode only. Their execution is similar in effect to that of the Load Clear E (D) Key Through Master Port command. The commands differ in that key bytes are initially decrypted using the electronic code book algorithm and the Master (M) Key register. Once decrypted, they are loaded byte-by-byte into the target key register, after having passed through the parity-check logic.

The command pending bit (S_6) is 1 during the entire decrypt-and-load operation. In addition, the busy bit (S_5) is 1 during the actual decryption process.

Load Clear IVE Register Through Master Port (85H)
Load Clear IVD Register Through Master Port (84H)

These commands are used in Multiplexed Control mode only. Their execution is virtually identical to that of the Load Clear E (or D) Key Through Master Port command. The commands differ in that the data written to the input register address is routed to either the Encryption Initializing Vector (IVE) or Decryption Initializing Vector (IVD) register instead of a key register. No parity checking occurs. The Command Pending bit (S_6) is 1 during the entire loading process.

Load Encrypted IVE Register Through Master Port (A5H).
Load Encrypted IVD Register Through Master Port (A4H).

These commands are analogous to the Load Encrypted E (or D) Key Through Master Port command. The data flow specifications set in the Mode register are overridden and the eight vector bytes are decrypted using the Decryption (D) Key register and the electronic code book algorithm. The resulting clear vector bytes are loaded into the target Initializing Vector register. No parity checking occurs. The Busy bit (S_5) does not become 1 during the decryption process, but the Command Pending bit (S_6) is 1 during the entire decryption-and-load operation.

Read Clear IVE Register Through Master Port (8DH).
Read Clear IVD Register Through Master Port (8CH).

In the Multiplexed Control mode, these commands override the data flow specifications set in the Mode register and connect the appropriate Initializing Vector register to the master port at the Output register address. In this state, each IV register appears as eight bytes of FIFO storage. The first byte of data is available six clocks after loading the Command register. The Command Pending bit in the Status register remains a 1 until sometime after the eighth byte is read out. The host system is responsible for reading exactly eight bytes.

Read Encrypted IVE Register Through Master Port (A9H).
Read Encrypted IVD Register Through Master Port (A8H).

In the Multiplexed Control mode only, these commands override the specifications set in the Mode register and encrypt the contents of the specified Initializing Vector register using the electronic code book algorithm and the Encrypt (E) key. The resulting cipher text is placed in the output register, where it can be read as eight bytes through the master port. During the actual encryption process, the Busy bit (S_5) is 1. When the Busy bit becomes 0, the encrypted vector bytes are ready to be read out. The Command Pending bit (S_6) is 1 during the entire encryption and output process; it becomes 0 when the eighth byte is read out. The host system is responsible for reading exactly eight bytes.

Encrypt with Master (M) Key (39H).

In the Multiplexed Control mode, this command overrides the data flow specifications set in the Mode register and causes the DCP to accept eight bytes from the master port, which are written to the Input register. When eight bytes have been received, the DCP encrypts

Commands (Continued)

the input using the Master (M) Key register. The encrypted data is loaded into the Output register, where it can be read out through the master port. The Command Pending bit (S_6) and the Busy (S_5) bit are used as status indicators in the three phases of this operation.

The Command Pending bit becomes 1 as soon as the Input register can accept data. When exactly eight bytes have been entered, the Busy bit becomes and remains 1 until the encryption process is complete. When Busy becomes 0, the encrypted data is available to be read out. The Command Pending bit returns to 0 when the eighth byte has been read.

Start Encryption (41H)
Start Decryption (40H)
Start (COH).

The three start commands begin normal data ciphering by setting the Status register's Start/Stop bit (S_7) to 1. The Start Encryption and Start Decryption commands explicitly specify the ciphering direction by forcing the Encrypt or Decrypt bit (M_4) in the Mode register to 1 or 0, respectively. The Start command, however, uses the current state of the Encrypt/Decrypt bit, as specified in a previous Mode register load.

When a start command has been entered, the port status flag ($\overline{MFLG}$ or $\overline{SFLG}$) associated with the Input register becomes active (Low), indicating that data may be written to the Input register to begin ciphering.

In Direct Control mode, the Start command is issued by raising the level on the AUX_5-$S/\overline{S}$ input (Table 4). The ciphering direction is specified by the level on AUX_6-$E/\overline{D}$. If AUX_6-$E/\overline{D}$ is High when AUX_5-$S/\overline{S}$ goes High, the command is Start Encryption; if AUX_6-$E/\overline{D}$ is Low, it is Start Decryption.

Stop (EOH).

The Stop command clears the Start/Stop bit (S_7) in the Status register. This action causes the input flag ($\overline{MFLG}$ or $\overline{SFLG}$) to become inactive and inhibits the loading of any further input into the algorithm unit. If ciphering is in progress [Busy bit (S_5) is 1 or AUX_2-$\overline{BSY}$ is active], it is allowed to finish, and any data in the Output register remains accessible.

In Direct Control mode, the Stop command is implied when the signal level on the AUX_5-$S/\overline{S}$ input goes from High to Low (Table 4).

Software Reset (00).

This command has the same effect as a hardware reset ($\overline{MAS}$ and $\overline{MDS}$ Low): it forces the DCP back to its default configuration, and all processing flags go into Inactive mode. The default configuration includes setting the Mode register to Electronic Code Book ciphering mode and establishes a dual-port configuration with master port clear and slave port encrypted.

Timing Requirements

The control and/or data signals and the timing requirements for clock/reset, Direct Control mode, Multiplexed Control mode (master port), master (slave) port read/write, and auxiliary port key entry functions are illustrated in Figures 8 through 12. The ac switching characteristics of the signals involved in the above functions are described in the AC Characteristics. The specific timing periods described are identified by numerics (1 through 48), which are referenced in both the timing diagrams and in the AC Characteristics.

A two-to-seven character symbol is listed in AC Characteristics for each period described. The symbol specifies the signal(s) involved, the state of each signal, and optionally, the port associated with a signal. Symbols are encoded as follows:

General Form: Ta Ab (Cb)

Where:

(1) T is a constant.

(2) a represents any one of the following symbols:

Symbol	*Meaning*
c	Clock
d	Delay
f	Fall Time
h	Hold Time
r	Rise Time
s	Setup Time
w	Width

(3) A,C represent any of the following signal names:

Symbol	*Signal Name*
A	Address Strobe
B	$\overline{BSY}$, Busy
C	Clock
D*	Data In or the address at the master port.
E	$E/\overline{D}$, Enable/Disable
F*	Flag ($\overline{MFLG}$, $\overline{SFLG}$, or ($\overline{AFLG}$)
G*	Data Strobe ($\overline{MDS}$, $\overline{SDS}$, or $\overline{ASTB}$)
K	$K/\overline{D}$, Key/Data
M	$C/\overline{K}$, Control/Key Mode
N	$S/\overline{S}$, Start/Stop
P	$\overline{PAR}$, Parity
Q*	Data Out (master or slave port)
R	$\overline{CP}$, Clock Pulse
S*	Chip Select (master or slave port)
W	$MR/\overline{W}$, Master Port read/write

Timing Requirements (Continued)

(4) b represents any one of the following signal state descriptors (symbol).

Symbol	*State Indicated*
h	High
l	Low
v	Valid
x	Invalid
z	High Impedance

*These signal names may be modified by the following optional numeric port identifiers:

Identifier	*Port*
1	Master Port
2	Slave Port
3	AUX (Key) Port

For example: D1 specifies data in at Master Port; F2 specifies Slave Port flag-SFLG.

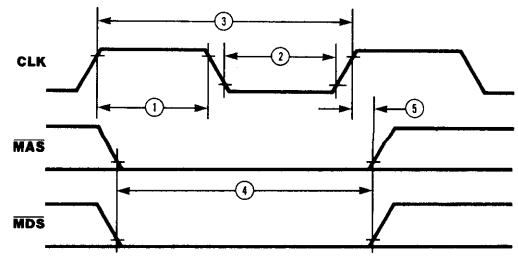


Figure 8. Clock and Reset

AC Switching Characteristics

Number	Symbol	Parameter	Min	Max	Notes*†
		Clock			
1	TwCh	Clock Width (High)	105		
2	TwCl	Clock Width (Low)	105		
3	TcC	Clock Cycle Time	250		
		Reset			
4	TdGll(Glh)	$\overline{MDS}$*$\overline{MAS}$ Low to $\overline{MDS}$*$\overline{MAS}$ High (Reset Pulse Width)	TC		
5	TdC(Glh)	Clock High to $\overline{MDS}$*$\overline{MAS}$ High	0	50	
		Direct Control Mode			
6	TsNl(Mh)	S/$\overline{S}$ Low to C/$\overline{K}$ High (Setup)	2TC		
7	TsKl(Mh)	K/$\overline{D}$ Low to C/$\overline{K}$ High (Setup)	2TC		
8	TdMh(Nh)	C/$\overline{K}$ High to S/$\overline{S}$ high	4TC		
9	TdMh(Kh)	C/$\overline{K}$ High to K/$\overline{D}$ High	4TC		
10	TsEv(Kh)	E/$\overline{D}$ Valid to K/$\overline{D}$ High (Setup)	2TC		
11	TdKh(Rl)	K/$\overline{D}$ High to $\overline{CP}$ Low		200	
12	ThKl(Ex)	K/$\overline{D}$ Low to E/$\overline{D}$ Invalid (Hold)	TC		
13	TdCl(Nh)	Clock Low to S/$\overline{S}$ Valid	20	80	
14	TsEv(Hn)	E/$\overline{D}$ Valid to S/$\overline{S}$ High (Setup)	2TC		
15	TdNh(Fll)	S/$\overline{S}$ High to $\overline{MFLG}$ ($\overline{SFLG}$) Low (Port Input Flag)		230	
16	TdCh(Fll)	Clock High to $\overline{MFLG}$ ($\overline{SFLG}$) Low (Port Input Flag)		230	1
17	TdCh(Bl)	Clock High to $\overline{BSY}$ Low		300	
18	TdCl(Bh)	Block Low to $\overline{BSY}$ High		220	
19	TdCh(Fll)	Clock High to $\overline{MFLG}$ ($\overline{SFLG}$) Low (Port Output Flag)		230	
20	TdNl(Flh)	S/$\overline{S}$ Low to $\overline{MFLG}$ ($\overline{SFLG}$) High (Port Input Flag)		230	2
		Multiplexed Control Mode – Master Port			
21	TwAl	$\overline{MAS}$ Width (Low)	80		
22	TdWv(Ah)	MR/$\overline{W}$ Valid to $\overline{MAS}$ High	40		
23	TsSll(Ah)	$\overline{MCS}$ Low to $\overline{MAS}$ High (Setup)	0		
24	ThAh(Slh)	$\overline{MAS}$ High to $\overline{MCS}$ High (Hold)	60		
25	TsDlv(Ah)	Address-In Valid to $\overline{MAS}$ High (Address Setup Time)	55		
26	ThAh(Dlx)	$\overline{MAS}$ High to Address-In Invalid (Address Hold Time)	60		

* Notes referenced at end of AC Characteristics table.

AC Switching Characteristics (Continued)

Number	Symbol	Parameter	Min	Max	Notes*†
		Master (Slave) Port Read/Write			
27	TdS1l(G1l)	$\overline{MCS}$ (SCS) Low to $\overline{MDS}$ ($\overline{SDS}$) Low	70		
28	ThG1h(S1h)	$\overline{MDS}$ ($\overline{SDS}$) High to $\overline{MCS}$ (SCS) High (Select-Hold Time)	0		3
29	TsWv(G1l)	MR/$\overline{W}$ Valid to $\overline{MDS}$ Low (Setup)	70		
30	ThG1h(Hwx)	$\overline{MDS}$ High to MR/$\overline{W}$ Invalid (Hold)	0		
31	TwG1l(G1h)	$\overline{MDS}$ ($\overline{SDS}$) Low to $\overline{MDS}$ ($\overline{SDS}$) High			
		Width—Write Data Read	125		
		Width—Status Register Read	155		
32	TdC1(G1h)	Clock Low to $\overline{MDS}$ ($\overline{SDS}$) High	20	70	
33	TdG1h(HG1l)	$\overline{MDS}$ ($\overline{SDS}$) High to $\overline{MDS}$ ($\overline{SDS}$) Low (Data Strobe Recovery Time)	125		
34	TsD1r(H1h)	Write-Data Valid to $\overline{MDS}$ ($\overline{SDS}$) High			
		Setup Time—Key Load	200		
		Setup Time—Data Write	100		
		Setup Time—Command/Mode Register Write	100		
35	ThG1h(D1x)	$\overline{MDS}$ ($\overline{SDS}$) High to Write-Data Invalid (Hold Time—All Writes)	40		
36	TdG1l(Q1v)	$\overline{MDS}$ ($\overline{SDS}$) Low to Read-Data Valid			
		Read Access Time—Status Register		155	
		Read Access Time—Data		120	
37	ThG1h(Q1x)	$\overline{MDS}$ ($\overline{SDS}$) High to Read-Data Invalid (Read Hold Time)	5	80	
38	TdG1l(F1h)	$\overline{MDS}$ ($\overline{SDS}$) Low to $\overline{MFLG}$ ($\overline{SFLG}$) High (Last Strobe)		125	4
39	TdG1l(Rh)	$\overline{MDS}$ High to $\overline{CP}$ High (Last Strobe, Key Load)		TC + 280	
40	ThG1(HN1)	$\overline{MDS}$ ($\overline{SDS}$) High to S/$\overline{S}$ Low (Hold Time After Last Input Strobe)	3TC		
41	TdG1(HPv)	$\overline{MDS}$ High to $\overline{PAR}$ Valid (Key Write)		200	

* Notes referenced at end of AC Characteristics table.

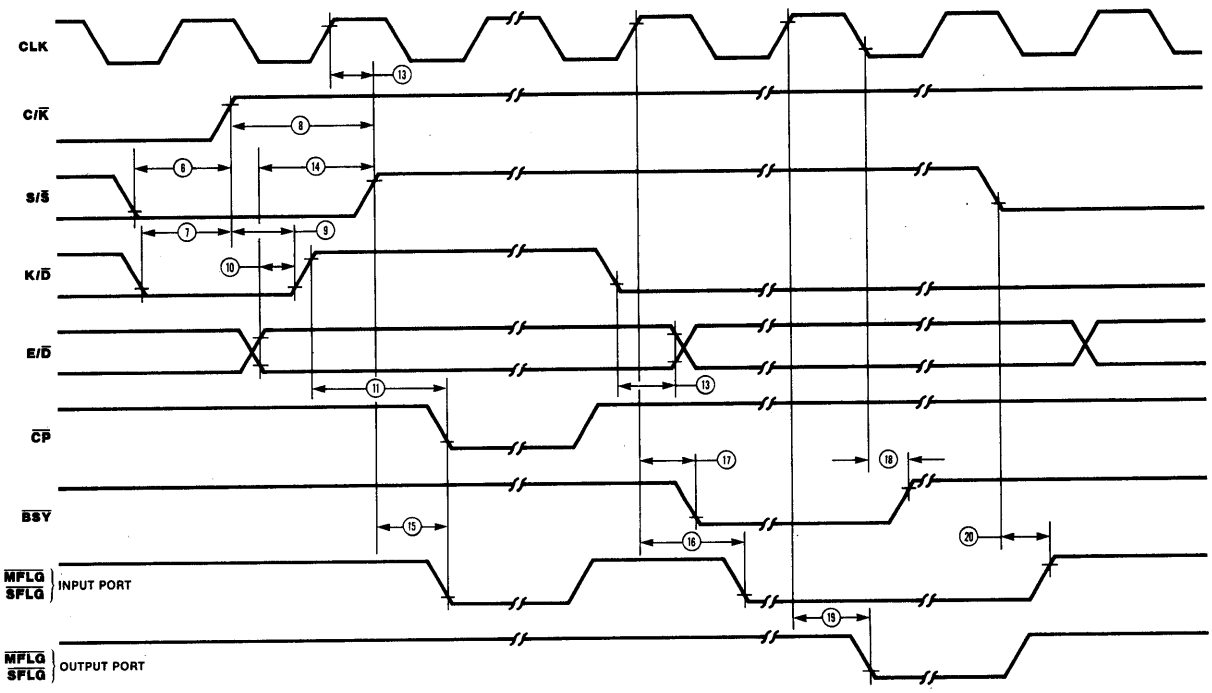


Figure 9. Control and Status Signals (Direct Control Mode)

AC Switching Characteristics (Continued)

Number	Symbol	Parameter	Min	Max	Notes*†
		Auxiliary Port Key Entry			
42	TwG3	$\overline{\text{ASTB}}$ Low to $\overline{\text{ASTB}}$ High (Width)	160		
43	TdCl(G3h)	Clock Low to $\overline{\text{ASTB}}$ High	20	70	
44	TdG3h(G31)	$\overline{\text{ASTB}}$ High to Next $\overline{\text{ASTB}}$ Low (Recovery Time)	125		
45	TsD3v(G3h)	Write-Data Valid to $\overline{\text{ASTB}}$ High (Data Setup Time)	200		
46	ThG3h(D3x)	$\overline{\text{ASTB}}$ High to Write-Data Invalid (Data Hold Time)	40		
47	TdG3h(Pr)	$\overline{\text{ASTB}}$ High to $\overline{\text{PAR}}$ Valid		200	
48	TdG31(F3h)	$\overline{\text{ASTB}}$ Low to $\overline{\text{AFLG}}$ High (Last Strobe)		230	

NOTES:
* All transition times are assumed to be ≤ 20 ns.
† All units in nanoseconds (ns). All timings are preliminary and subject to change.
1. Parameter TaCh(F11) applies to all input blocks except the first (when S/$\overline{\text{S}}$ first goes High).
2. When S/$\overline{\text{S}}$ goes inactive (Low) in Direct Control mode, the flag associated with the input port turns off.
3. Direct Control mode only.
4. In Cipher Feedback mode, the port flag ($\overline{\text{MFLG}}$ or $\overline{\text{SFLG}}$) goes inactive following the leading edge of the first data strobe ($\overline{\text{MDS}}$ or $\overline{\text{SDS}}$); in all other modes and operations, the flags go inactive on the eighth data strobe.

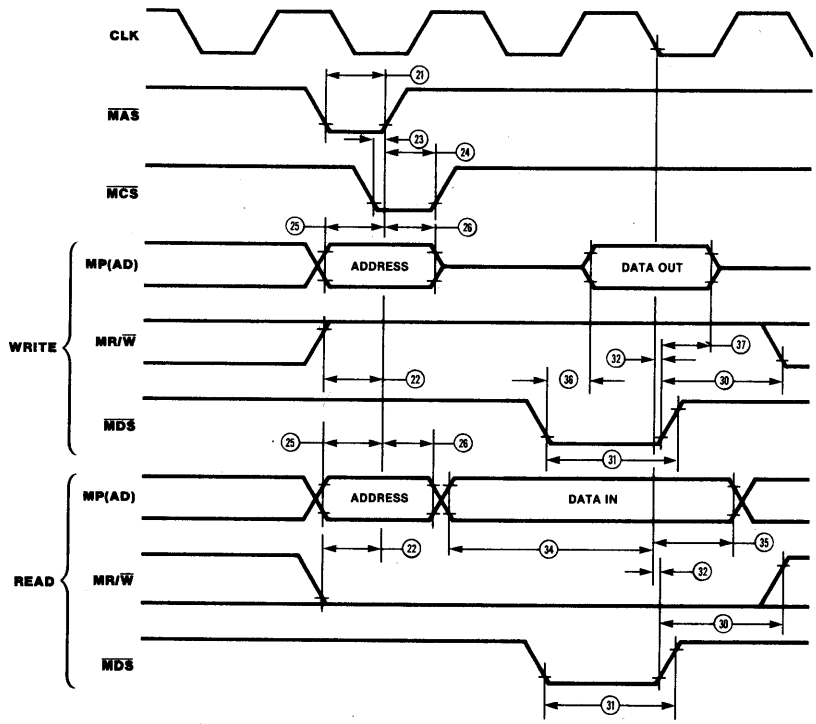


Figure 10. Master Port, Multiplexed Control Mode Read/Write Timing

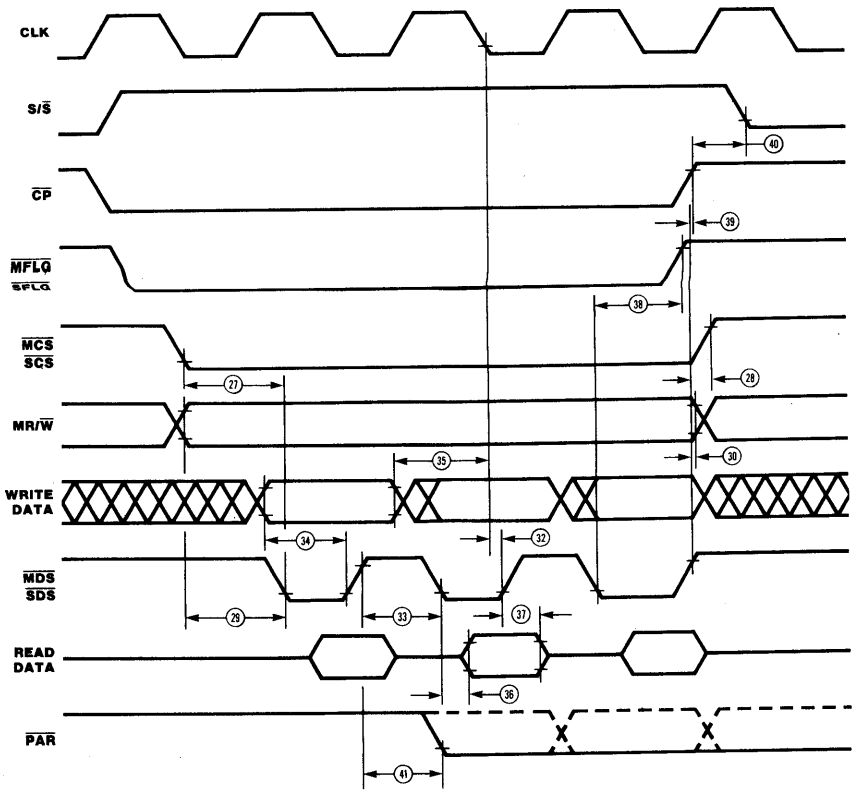


Figure 11. Master (Slave) Port Read/Write

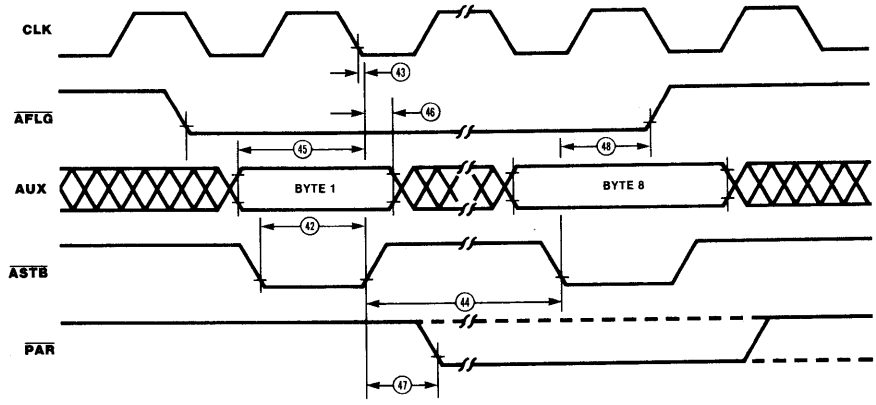


Figure 12. Auxiliary Port Key Entry

Ordering Information

Product Number	Package/ Temp.	Speed	Description	Product Number	Package/ Temp.	Speed	Description
Z8068	DS	4.0 MHz	DCP (40-pin)				

NOTES: D = Cerdip, S = 0°C to 70°C.

Universal Peripherals

Zilog

Pioneering the Microworld

Universal Peripherals

Two Versions Extend Range of Applications

September 1983

Zilog's Universal Peripheral Components Family is more than a group of simple I/O circuits—they are intelligent, fully programmable devices capable of performing complicated tasks independently. Their capabilities unburden the master CPU, reduce bus traffic, increase system throughput, and greatly simplify overall system hardware design requirements.

The peripheral components, where needed, are produced in two versions to increase their range of application. One version, identified by the number Z80xx, is capable of interfacing with Zilog's multiplexed Z-BUS only or with both the Z-BUS and conventional multiplexed buses. The second version, identified by the number Z85xx, is capable of interfacing with conventional nonmultiplexed buses. Many of these Z85xx peripherals will function with and add capability to non-Zilog CPUs. Contact your local Zilog sales office, local distributor or representative for additional information and detailed specifications. This section of the data book includes only product specifications or product briefs on the Z85xx series of components. For the specifications or briefs on the Z80xx components refer to the Z8000 peripherals section.

All of the peripheral components are extensively programmable to permit each to be tailored to its own application(s). All Z-BUS peripherals share common interrupt and bus-request structures; they can also be operated in either a priority-interrupt or polled environment.

Counting, timing, and parallel I/O transfer problems are easily solved using the **Z8036/Z8536 CIO Counter/Timer and I/O Unit.** This component has three 16-bit counter/timers, three I/O ports, and can double as a programmable priority-interrupt controller.

Data communications problems are neatly handled by the **Z8030/Z8530 SCC Serial Communications Controller**. This device is a serial, dual-channel, multiprotocol controller which supports all popular communications formats. The SCC supports virtually all serial data transfer applications.

Interface problems with the interconnection of major components within an asynchronous, parallel processor system can be solved using the **Z8038 Z-FIO FIFO I/O Interface Unit.** This general-purpose interface unit provides expandable, bidirectional buffering between asynchronous CPUs in a parallel processing network, or between a CPU and peripheral circuits and/or devices. The Z-FIO can be used with systems having either multiplexed or nonmultiplexed buses.

General-purpose control and data manipulation problems are easily handled by the **Z8090/4 and Z8590/4 UPC Universal Peripheral Controller.** The UPC is a complete microcomputer designed for off-line applications. This microcomputer executes the same friendly, capable instruction set as Zilog's Z8 microcomputer; it has three I/O ports, six levels of priority-interrupt, and 2K bytes of memory on chip. The UPC is intended for applications that require an intelligent peripheral controller that can assume many of the tasks normally required of the master CPU.

Two new universal peripherals have been added to the ever expanding line of Zilog peripherals. They are the **Z8581 Clock Generator and Controller (CGC)** and the **Z8531 ASCC Asynchronous Serial Communications Controller.**

The **Z8581 Clock Generator and Controller (CGC)** is a versatile addition to Zilog's family of universal microprocessor components. The selective clock-stretching capabilities and variety of timing outputs of this device allow it to meet the timing design requirements of various microprocessors easily, including those of LSI and VLSI peripherals.

The outputs of the Z8581 CGC directly drive the Z80 and Z8000 microprocessor clock inputs. The oscillator input frequency reference sources can be either crystals or TTL-compatible oscillators.

To complement the Z8530/Z8030 SCC Serial Communications Controller, Zilog has introduced an asynchronous version designated the Z8531 ASCC. It features two independent 0 to 1M bit/second, full-duplex channels—each with a separate crystal oscillator, baud rate generator and digital phase-locked loop for clock recovery. The Z8581 ASCC has programmable NRZ, NRZI, or FM data encoding.

The LSI and VLSI components now available can meet the design needs of today, while Zilog *continues to design state-of-the-art devices for the needs of tomorrow.*

Z8530 SCC Serial Communications Controller

Zilog

Product Specification

September 1983

Features

- Two independent, 0 to 1M bit/second, full-duplex channels, each with a separate crystal oscillator, baud rate generator, and Digital Phase-Locked Loop for clock recovery.
- Multi-protocol operation under program control; programmable for NRZ, NRZI, or FM data encoding.
- Asynchronous mode with five to eight bits and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Synchronous mode with internal or external character synchronization on one or two synchronous characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and SDLC Loop mode operation.
- Local Loopback and Auto Echo modes.

General Description

The Z8530 SCC Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with conventional non-multiplexed buses. The SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, Digital Phase-Locked Loops, and crystal oscillators that dramatically reduce the need for external logic.

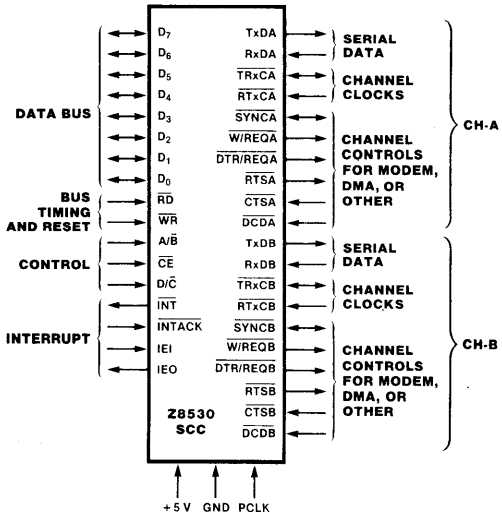


Figure 1. Pin Functions

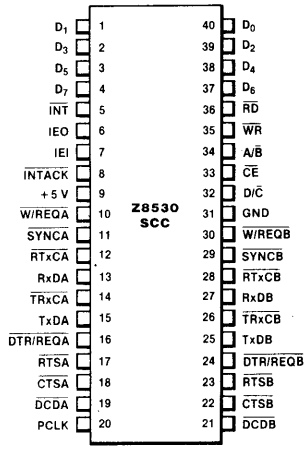


Figure 2. Pin Assignments

General Description (Continued)

The SCC handles asynchronous formats, Synchronous byte-oriented protocols such as IBM Bisync, and Synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (cassette, diskette, tape drives, etc.).

The device can generate and check CRC codes in any Synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-Bus daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

The Z8530 SCC is packaged in a 40-pin ceramic DIP and uses a single +5 V power supply.

Pin Description

The following section describes the pin functions of the SCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

A/$\overline{B}$. *Channel A/Channel B Select* (input). This signal selects the channel in which the read or write operation occurs.

$\overline{CE}$. *Chip Enable* (input, active Low). This signal selects the SCC for a read or write operation.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

D/$\overline{C}$. *Data/Control Select* (input). This signal defines the type of information transferred to or from the SCC. A High means data is transferred; a Low indicates a command.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accomodate slow rise-time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

D_0-D_7. *Data Bus* (bidirectional, 3-state). These lines carry data and commands to and from the SCC.

$\overline{DTR}$/$\overline{REQA}$, $\overline{DTR}$/$\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the SCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the SCC interrupt daisy chain settles. When $\overline{RD}$ becomes active, the SCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of PCLK.

PCLK. *Clock* (input). This is the master SCC clock used to synchronize internal signals PCLK is a TTL level signal.

$\overline{RD}$. *Read* (input, active Low). This signal indicates a read operation and when the SCC is selected, enables the SCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the SCC is the highest priority device requesting an interrupt.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RTxCA}$, $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{SYNC}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 11) is set, the $\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto

Pin Description (Continued)

Enable is on, the signal goes High after the transmitter is empty. In Synchronous mode or in Asynchronous mode with Auto Enable off, the RTS pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

SYNCA, SYNCB. *Synchronization* (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 (Figure 10) but have no other function.

In External Synchronization mode with the crystal oscillator not selected, these lines also act as inputs. In this mode, $\overline{SYNC}$ must be driven Low two receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of $\overline{SYNC}$.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the receive clock cycle in which synchronous characters are recognized. The synchronous condition is not latched, so these outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, these pins act as outputs and are valid on receipt of a flag.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

TRxCA, TRxCB. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{TRxC}$ may supply the receive clock or the transmit clock in the input mode or supply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

WR. *Write* (input, active Low). When the SCC is selected, this signal indicates a write operation. The coincidence of $\overline{RD}$ and $\overline{WR}$ is interpreted as a reset.

W/REQA, W/REQB. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait.

Functional Description

The functional capabilities of the SCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, the SCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities. The SCC provides two independent full-duplex channels programmable for use in any common Asynchronous or Synchronous data-communication protocol. Figure 3 and the following description briefly detail these protocols.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection

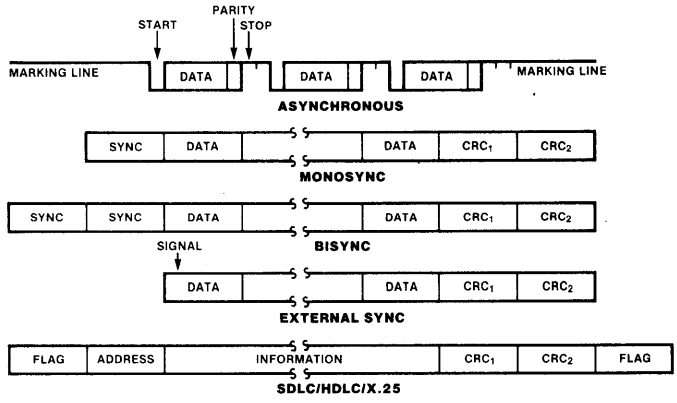


Figure 3. Some SCC Protocols

Functional Description (Continued)

mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing or error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The SCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs. In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed as an input used for functions such as monitoring a ring indicator.

Synchronous Modes. The SCC supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes, allowing character synchronization with a 6-bit or 8-bit synchronous character (Monosync), any 12-bit synchronization pattern (Bisync), or with an external synchronous signal. Leading sync characters can be removed without interrupting the CPU.

Five- or 7-bit synchronous characters are detected with 8- or 16-bit patterns in the SCC by overlapping the larger pattern across multiple incoming synchronous characters as shown in Figure 4.

CRC checking for Synchronous byte-oriented modes is delayed by one character time so that the CPU may disable CRC checking on specific characters. This permits the implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16} + X^{15} + X^2 + 1$) and CCITT ($X^{16} + X^{12} + X^5 + 1$) error checking polynomials are supported. Either polynomial may be selected in all Synchronous modes. Users may preset the CRC generator and checker to all 1s or all 0s. The SCC also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows for high speed transmissions under DMA control, with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 6-, 8-, or 16-bit synchronous characters, regardless of the programmed character length.

The SCC supports Synchronous bit-oriented protocols, such as SDLC and HDLC, by performing automatic flag sending, zero insertion, and CRC generation. A special command can be used to abort a frame in transmission. At the end of a message, the SCC automatically transmits the CRC and trailing flag when the transmitter underruns. The transmitter may also be programmed to send an idle line consisting of continuous flag characters or a steady marking condition.

If a transmit underrun occurs in the middle of a message, an external/status interrupt warns the CPU of this status change so that an abort may be issued. The SCC may also be programmed to send an abort itself in case of an underrun, relieving the CPU of this task. One to eight bits per character can be sent, allowing reception of a message with no prior information about the character structure in the information field of a frame.

The receiver automatically acquires synchronization on the leading flag of a frame in SDLC or HDLC and provides a synchronization signal on the $\overline{\text{SYNC}}$ pin (an interrupt can also be programmed). The receiver can be programmed to search for frames addressed by a single byte (or four bits within a byte) of a user-selected address or to a global broadcast address. In this mode, frames not matching either the user-selected or broadcast address are ignored. The number of address bytes can be extended under software control. For receiving data, an interrupt on the first received character, or an interrupt on every character, or on special condition only (end-of-frame) can be selected. The receiver automatically deletes all 0s inserted by the transmitter during character assembly. CRC is also calculated and is automatically checked to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. In SDLC mode, the SCC must be programmed to use the SDLC CRC polynomial, but the generator and checker may be preset to all 1s or all 0s.

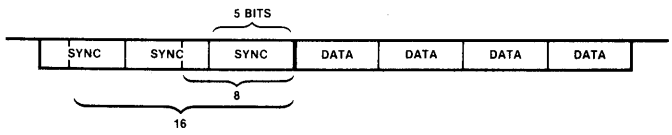


Figure 4. Detecting 5- or 7-Bit Synchronous Characters

Functional Description (Continued)

The CRC is inverted before transmission and the receiver checks against the bit pattern 0001110100001111.

NRZ, NRZI or FM coding may be used in any 1x mode. The parity options available in Asynchronous modes are available in Synchronous modes.

The SCC can be conveniently used under DMA control to provide high speed reception or transmission. In reception, for example, the SCC can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The SCC then issues an end-of-frame interrupt and the CPU can check the status of the received message. Thus, the CPU is freed for other service while the message is being received. The CPU may also enable the DMA first and have the SCC interrupt only on end-of-frame. This procedure allows all data to be transferred via the DMA.

SDLC Loop Mode. The SCC supports SDLC Loop mode in addition to normal SDLC. In an SDLC Loop, there is a primary controller station that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the SCC performs the functions of a secondary station while an SCC operating in regular SDLC mode can act as a controller (Figure 5).

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay. The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End Of Poll), around the loop. The EOP character is the bit pattern 11111110. Because of zero insertion during messages, this bit pattern is unique and easily recognized.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag sequence. The secondary station now places its message on the loop and terminates the message with an EOP. Any secondary stations further down the loop with messages to transmit can then append their messages to the message of the first secondary station by the same process. Any secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop (except upon recognizing an EOP).

SDLC Loop mode is a programmable option in the SCC. NRZ, NRZI, and FM coding may all be used in SDLC Loop mode.

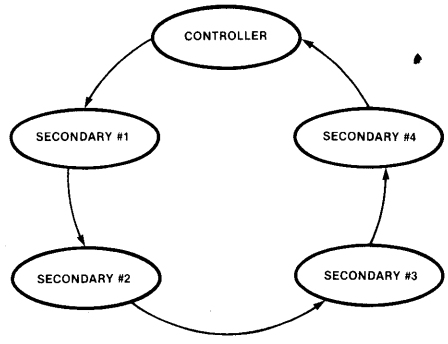


Figure 5. An SDLC Loop

Baud Rate Generator. Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{\text{TRxC}}$ pin, the output of the baud rate generator may be echoed out via the $\overline{\text{TRxC}}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds):

$$\text{baud rate} = \frac{1}{2\,(\text{time constant} + 2) \times (\text{BR clock period})}$$

Digital Phase-Locked Loop. The SCC contains a Digital Phase-Locked-Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the SCC receive clock, the transmit clock, or both.

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the

Functional Description (Continued)

incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the SCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding. The SCC may be programmed to encode and decode the serial data in four different ways (Figure 6). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the SCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

Auto Echo and Local Loopback. The SCC is capable of automatically echoing everything it receives. This feature is useful mainly in Asynchronous modes, but works in Synchronous and SDLC modes as well. In Auto Echo mode, TxD is RxD. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The SCC is also capable of local loopback. In this mode TxD is RxD, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works in Asynchronous, Synchronous and SDLC modes with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The SCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the SCC are automatically updated whenever any function is performed. For example, end-of-frame in SDLC mode sets a bit in one of these status registers. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be

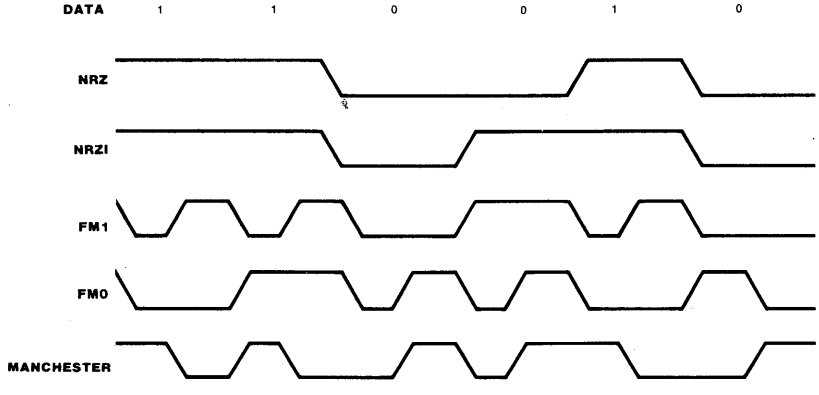


Figure 6. Data Encoding Methods

Functional Description (Continued)

read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. When an SCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 10 and 11).

To speed interrupt response time, the SCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the SCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the interrupt priority chain (Figure 7). As a microprocessor peripheral, the SCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the data bus.

In the SCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the SCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the SCC and external to the SCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the SCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is one of the following: receiver overrun, framing error in Asynchronous mode, end-of-frame in SDLC mode and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$, and $\overline{SYNC}$ pins; however, an

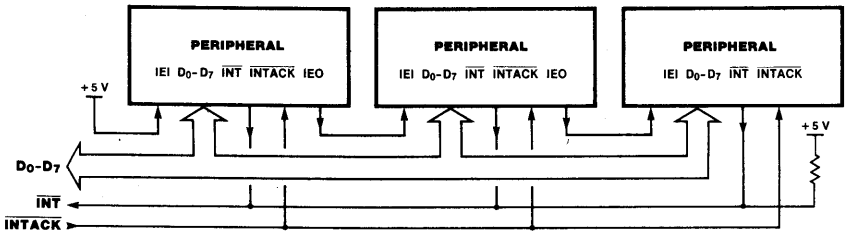


Figure 7. Interrupt Schedule

Functional Description (Continued)

External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break (Asynchronous mode), Abort (SDLC mode) or EOP (SDLC Loop mode) sequence in the data stream. The interrupt caused by the Abort or EOP has a special feature allowing the SCC to interrupt when the Abort or EOP sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and the accurate timing of the Abort condition in external logic in SDLC mode. In SDLC Loop mode, this feature allows secondary stations to recognize the wishes of the primary station to regain control of the loop during a poll sequence.

CPU/DMA Block Transfer. The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}$/$\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}$/$\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the SCC $\overline{\text{REQUEST}}$ output indicates that the SCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the SCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}$/$\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The SCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 8).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, two sync-character (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a

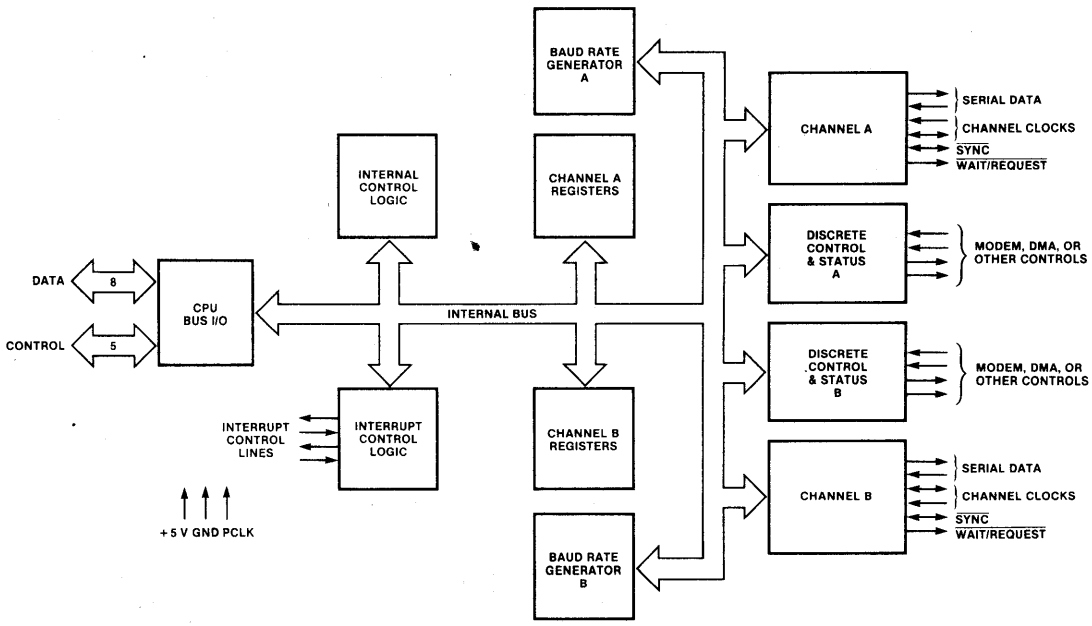


Figure 8. Block Diagram of SCC Architecture

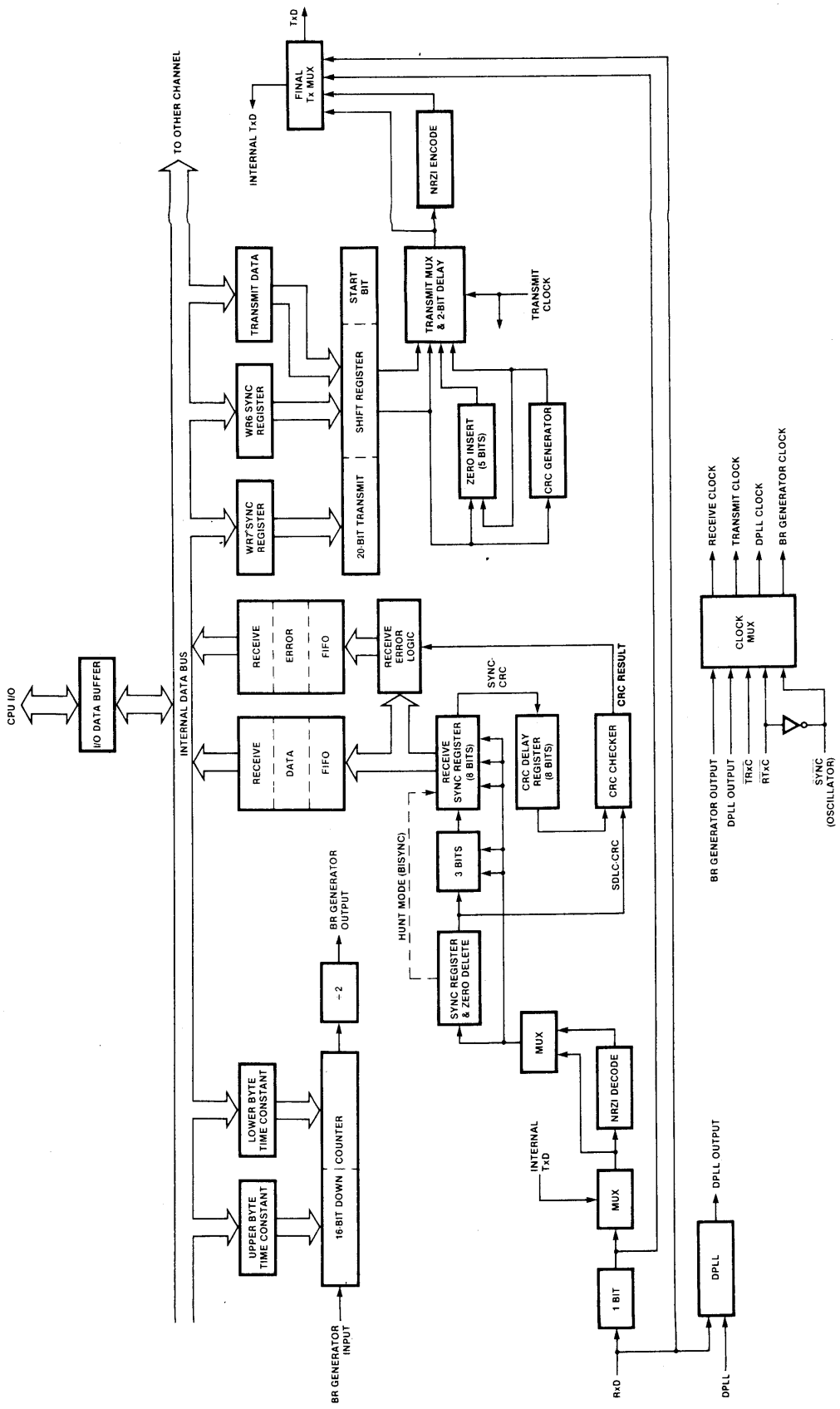


Figure 9. Data Path

Architecture (Continued)

write only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0-WR15 — Write Registers 0 through 15.

RR0-RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The SCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 9 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode (the character length in Asynchronous modes also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and a 20-bit Transmit Shift register that can be loaded either from the synchronous character registers or from the Transmit Data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data output (TxD)

Read Register Functions	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	Upper byte of baud rate generator time constant
RR15	External/Status interrupt information
Write Register Functions	
WR0	CRC initialize, initialization commands for the various modes, Register Pointers
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync characters or SDLC address field
WR7	Sync character or SDLC flag
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The SCC contains 13 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

In the SCC, register addressing is direct for the data registers only, which are selected by a High on the D/$\overline{C}$ pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed. All of the registers in the SCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

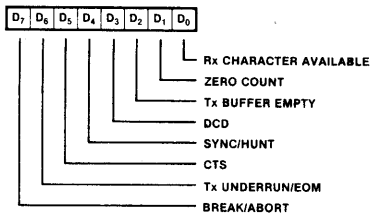
The system program first issues a series of commands to initialize the basic mode of operation. This is followed by other commands to qualify conditions within the selected mode. For example, the Asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

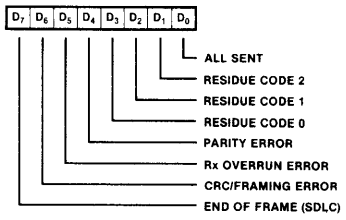
Read Registers. The SCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 10 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

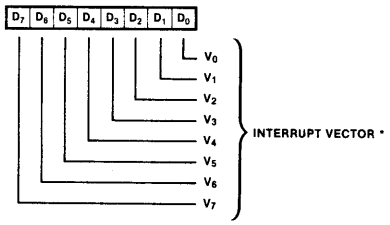
Read Register 0



Read Register 1

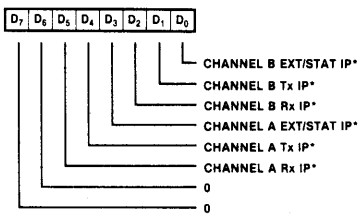


Read Register 2



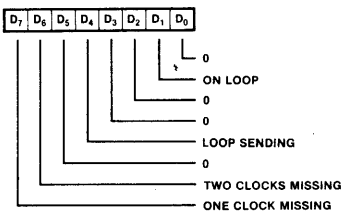
*MODIFIED IN B CHANNEL

Read Register 3

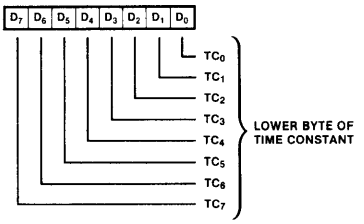


*ALWAYS 0 IN B CHANNEL

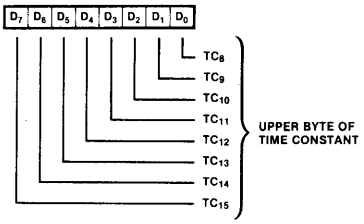
Read Register 10



Read Register 12



Read Register 13



Read Register 15

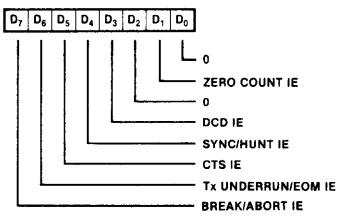
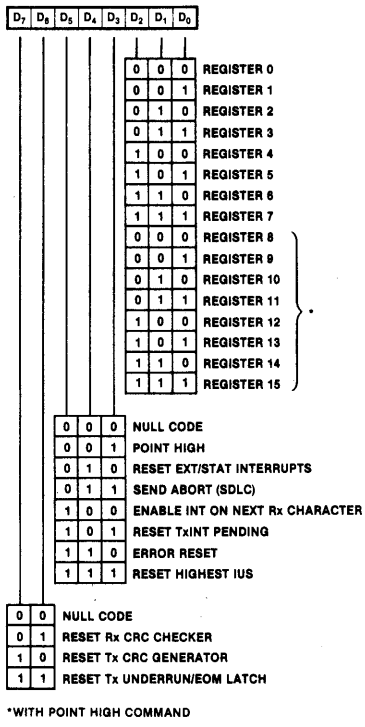


Figure 10. Read Register Bit Functions

Programming (Continued)

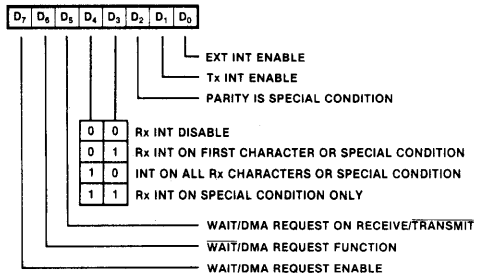
Write Registers. The SCC contains 13 write registers (14 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 11 shows the format of each write register.

Write Register 0

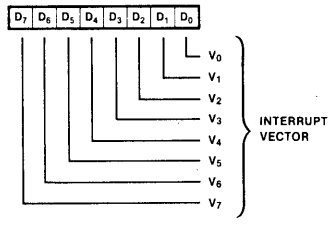


*WITH POINT HIGH COMMAND

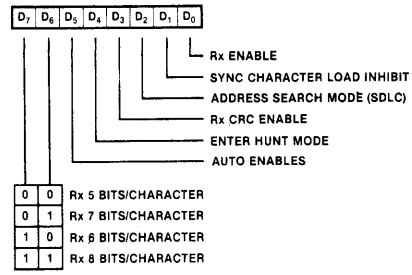
Write Register 1



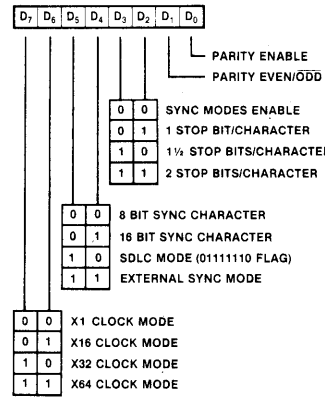
Write Register 2



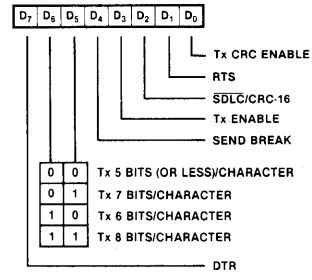
Write Register 3



Write Register 4



Write Register 5



Write Register 6

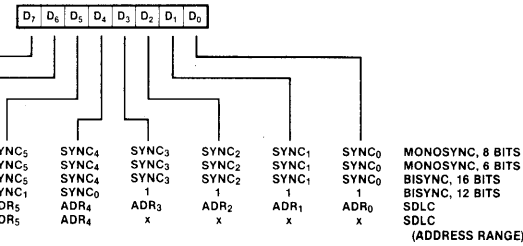
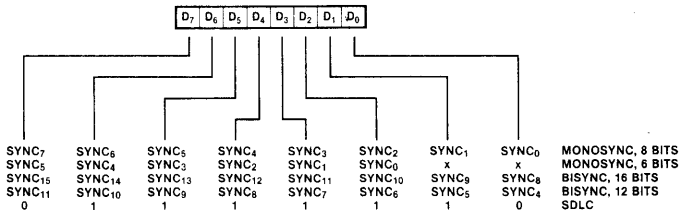
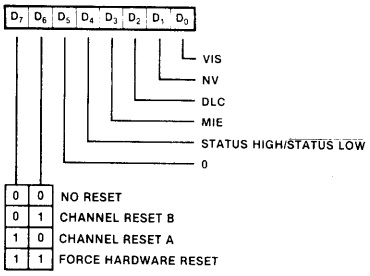


Figure 11. Write Register Bit Functions

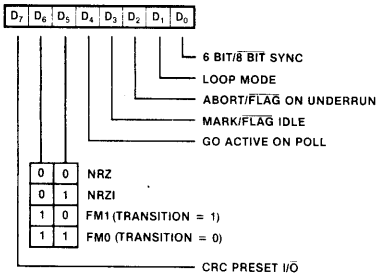
Write Register 7



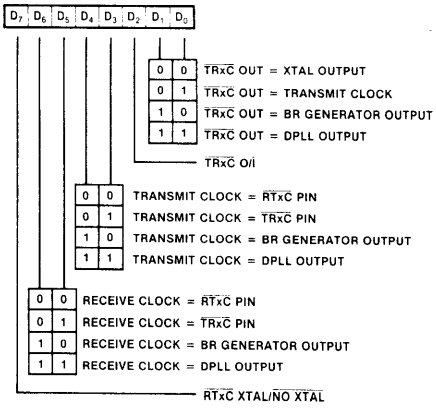
Write Register 9



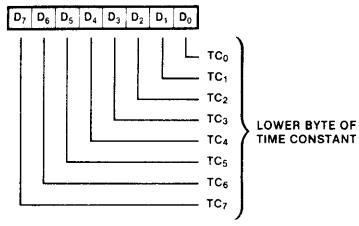
Write Register 10



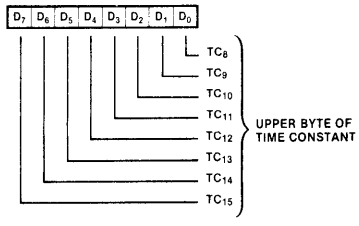
Write Register 11



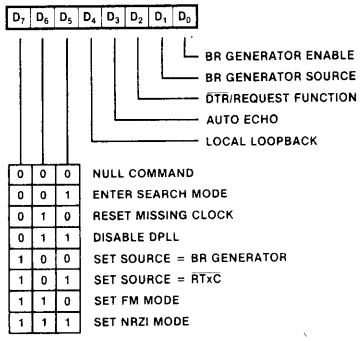
Write Register 12



Write Register 13



Write Register 14



Write Register 15

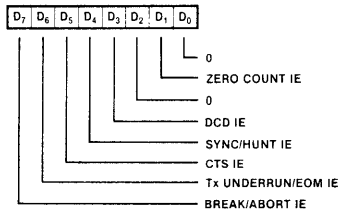


Figure 11. Write Register Bit Functions (Continued)

Timing

The SCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to PCLK. Since PCLK has no phase relationship with $\overline{WR}$ and $\overline{RD}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the SCC. The recovery time required for proper operation is specified from the rising edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the SCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the SCC. This time must be at least 6 PCLK cycles plus 200 ns.

Read Cycle Timing. Figure 12 illustrates Read cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{RD}$ falls or if it rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

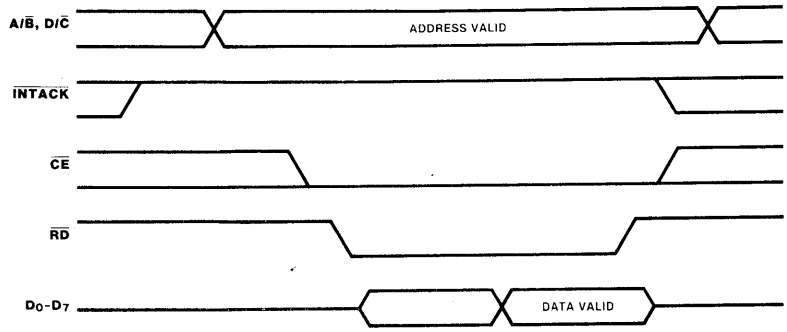


Figure 12. Read Cycle Timing

Write Cycle Timing. Figure 13 illustrates Write cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{WR}$ falls or if it rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened.

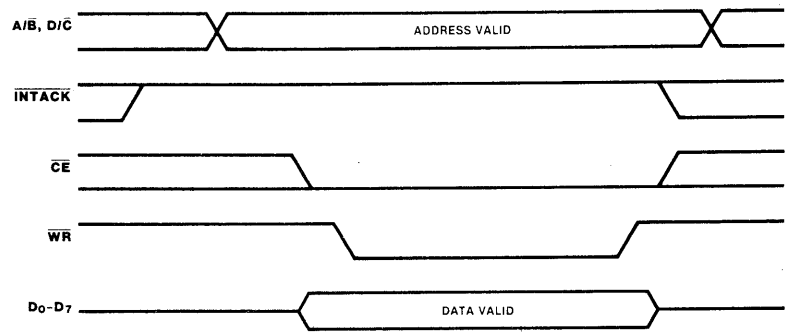


Figure 13. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 14 illustrates Interrupt Acknowledge cycle timing. Between the time $\overline{INTACK}$ goes Low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending in the SCC and IEI is High when $\overline{RD}$ falls, the Acknowledge cycle is intended for the SCC. In this case, the SCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0-D_7 and it then sets the appropriate Interrupt-Under-Service latch internally.

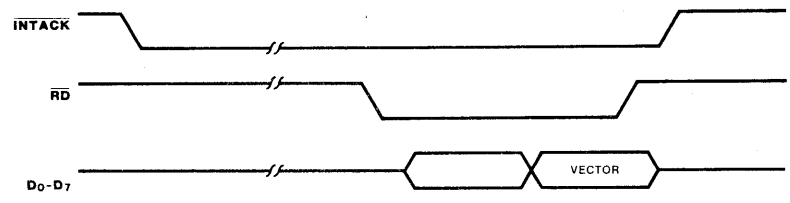


Figure 14. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

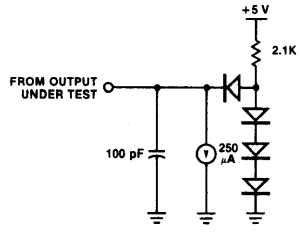


Figure 15. Standard Test Load

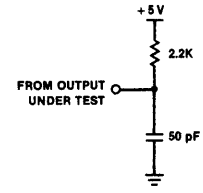


Figure 16. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10.0	µA	0.4 ≤ V_{IN} ≤ +2.4V
I_{OL}	Output Leakage		±10.0	µA	0.4 ≤ V_{OUT} ≤ +2.4V
I_{CC}	V_{CC} Supply Current		250	mA	

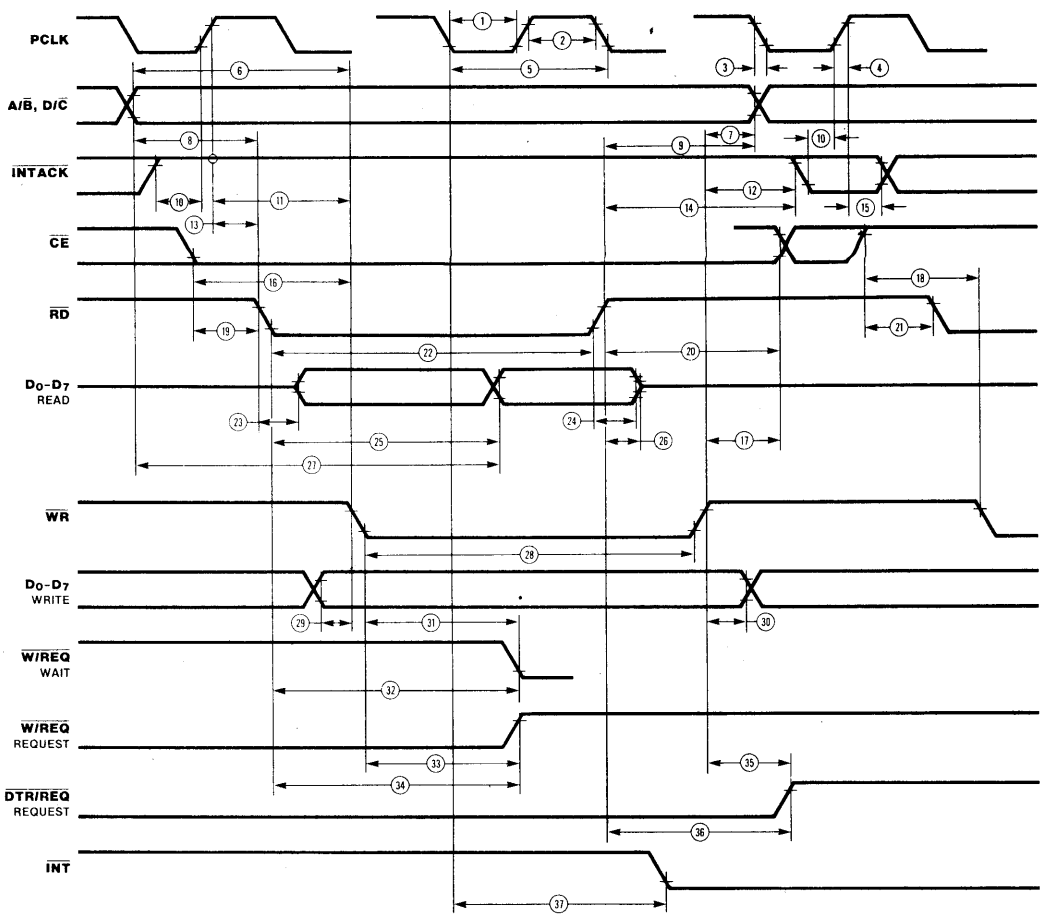
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

Read and Write Timing

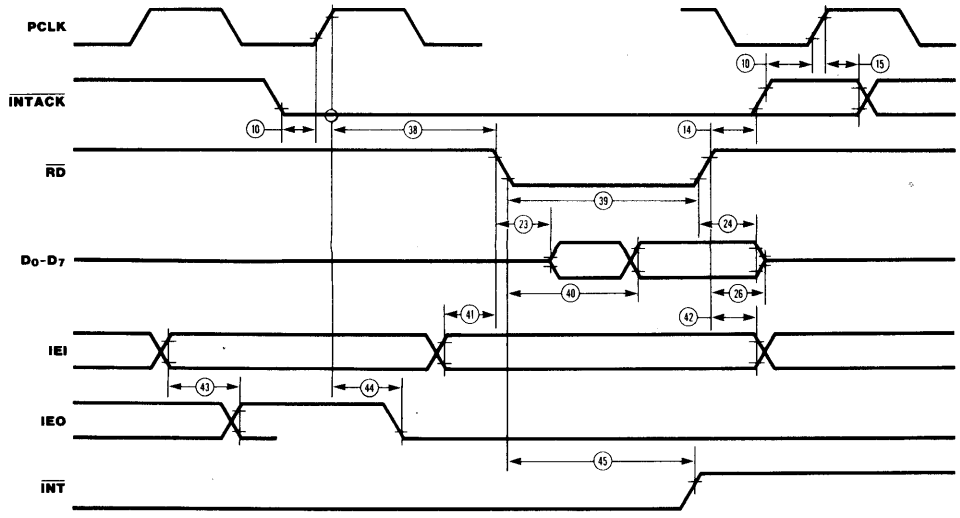


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwPCl	PCLK Low Width	105	2000	70	1000	
2	TwPCh	PCLK High Width	105	2000	70	1000	
3	TfPC	PCLK Fall Time		20		10	
4	TrPC	PCLK Rise Time		20		15	
5	TcPC	PCLK Cycle Time	250	4000	165	2000	
6	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		80		
7	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		
8	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		80		
9	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		
10	TslA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	0		0		
11	TslAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		160		1
12	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		
13	TslAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		160		1
14	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		
15	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	100		100		
16	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		
17	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		
18	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		70		
19	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		1
20	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		1
21	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		70		1
22	TwRDl	$\overline{RD}$ Low Width	390		250		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
24	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
25	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		250		180	
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70		45	2

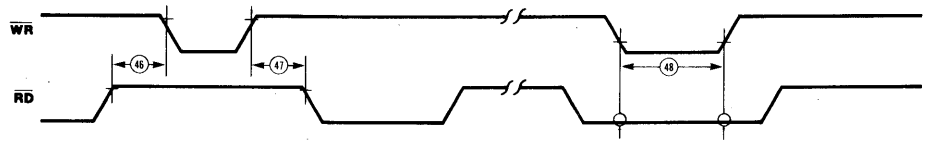
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a ±0.5 V change in the output with a maximum dc load and minimum ac load.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

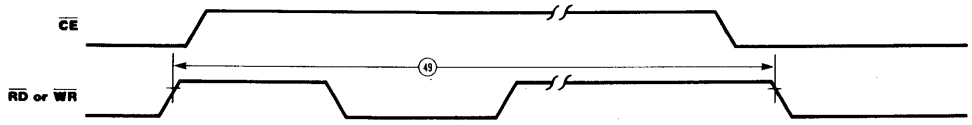
Interrupt Acknowledge Timing



Reset Timing



Cycle Timing

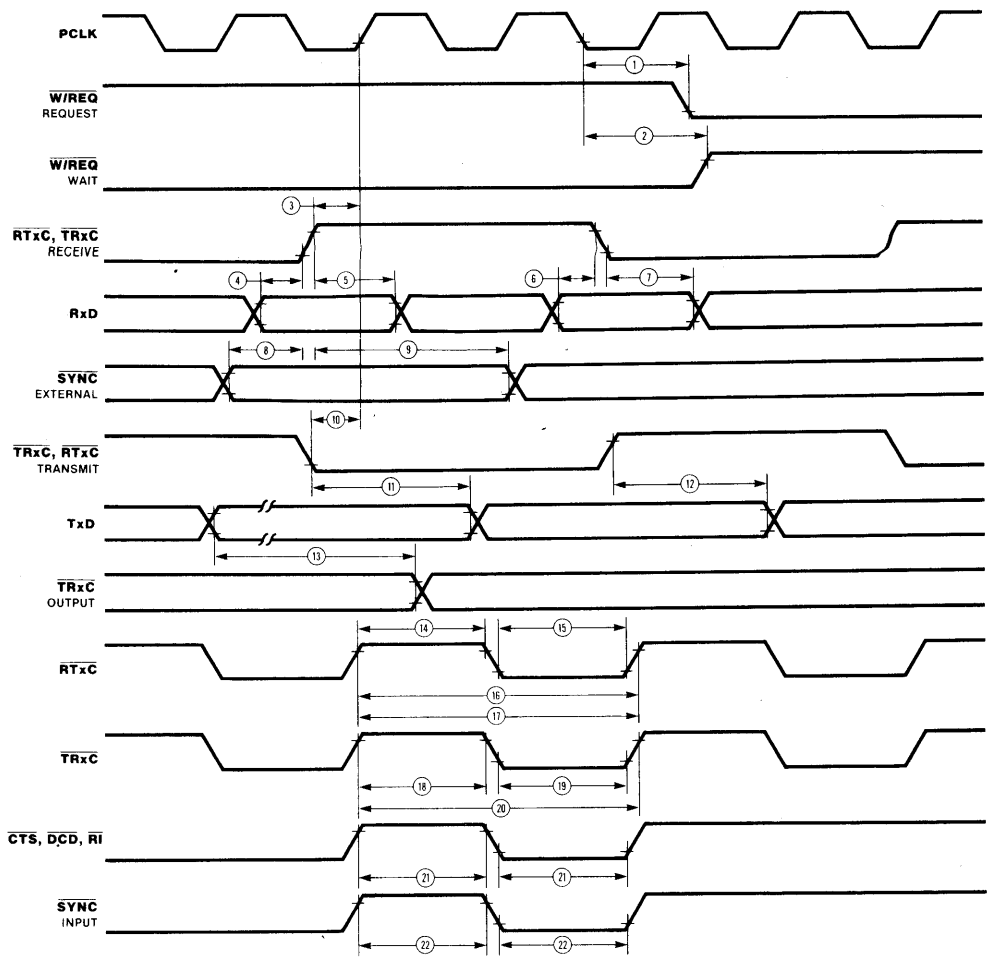


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
27	TdA(DR)	Address Required Valid to Read Data Valid Delay		590		420	
28	TwWR1	WR Low Width	390		250		
29	TsDW(WR)	Write Data to WR ↓ Setup Time	0		0		
30	ThDW(WR)	Write Data to WR ↑ Hold Time	0		0		
31	TdWR(W)	WR ↓ to Wait Valid Delay		240		200	4
32	TdRD(W)	RD ↓ to Wait Valid Delay		240		200	4
33	TdWRf(REQ)	WR ↓ to W/REQ Not Valid Delay		240		200	
34	TdRDf(REQ)	RD ↓ to W/REQ Not Valid Delay		240		200	
35	TdWRr(REQ)	WR ↑ to DTR/REQ Not Valid Delay		5TcPC +300		5TcPC +250	
36	TdRDr(REQ)	RD ↑ to DTR/REQ Not Valid Delay		5TcPC +300		5TcPC +250	
37	TdPC(INT)	PCLK ↓ to INT Valid Delay		500		500	4
38	TdIAi(RD)	INTACK to RD ↓ (Acknowledge) Delay	250		250		5
39	TwRDA	RD (Acknowledge) Width	285		250		
40	TdRDA(DR)	RD ↓ (Acknowledge) to Read Data Valid Delay		190		180	
41	TsIEI(RDA)	IEI to RD ↓ (Acknowledge) Setup Time	120		100		
42	ThIEI(RDA)	IEI to RD ↑ (Acknowledge) Hold Time	0		0		
43	TdIEI(IEO)	IEI to IEO Delay Time		120		100	
44	TdPC(IEO)	PCLK ↑ to IEO Delay		250		250	
45	TdRDA(INT)	RD ↓ to INT Inactive Delay		500		500	4
46	TdRD(WRQ)	RD ↑ to WR ↓ Delay for No Reset	30		15		
47	TdWRQ(RD)	WR ↑ to RD ↓ Delay for No Reset	30		30		
48	TwRES	WR and RD Coincident Low for Reset	250		250		
49	Trc	Valid Access Recovery Time	6TcPC +200		6TcPC +130		3

NOTES:
3. Parameter applies only between transactions involving the SCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any SCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the SCC, and TdIEIf(IEO) for each device separating them in the daisy chain.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

General Timing

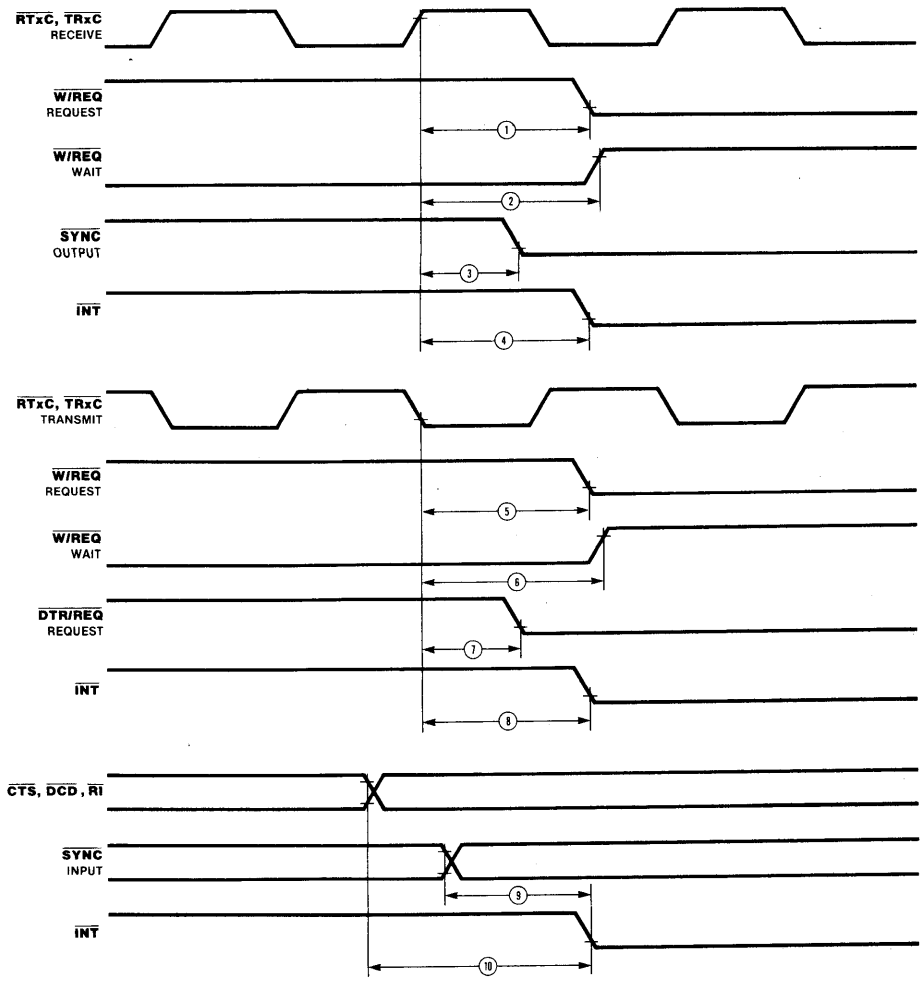


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdPC(REQ)	PCLK ↓ to W/REQ Valid Delay		250		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350		350	
3	TsRXC(PC)	RxC ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	80	TwPC1	70	TwPC1	1,4
4	TsRXD(RXCr)	RxD to RxC ↑ Setup Time (X1 Mode)	0		0		1
5	ThRXD(RXCr)	RxD to RxC ↑ Hold Time (X1 Mode)	150		150		1
6	TsRXD(RXCf)	RxD to RxC ↓ Setup Time (X1 Mode)	0		0		1,5
7	ThRXD(RXCf)	RxD to RxC ↓ Hold Time (X1 Mode)	150		150		1,5
8	TsSY(RXC)	SYNC to RxC ↑ Setup Time	-200		-200		1
9	ThSY(RXC)	SYNC to RxC ↑ Hold Time	3TcPC +200		3TcPC +200		1
10	TsTXC(PC)	TxC ↓ to PCLK ↑ Setup Time	0		0		2,4
11	TdTXCf(TXD)	TxC ↓ to TxD Delay (X1 Mode)		300		230	2
12	TdTXCr(TXD)	TxC ↑ to TxD Delay (X1 Mode)		300		230	2,5
13	TdTXD(TRX)	TxD to TRxC Delay (Send Clock Echo)		200		200	
14	TwRTXh	RTxC High Width	180		180		6
15	TwRTXl	RTxC Low Width	180		180		6
16	TcRTX	RTxC Cycle Time	400		400		6
17	TcRTXX	Crystal Oscillator Period	250	1000	250	1000	3
18	TwTRXh	TRxC High Width	180		180		6
19	TwTRXl	TRxC Low Width	180		180		6
20	TcTRX	TRxC Cycle Time	400		400		6
21	TwEXT	DCD or CTS Pulse Width	200		200		
22	TwSY	SYNC Pulse Width	200		200		

NOTES:
1. RxC is RTxC or TRxC, whichever is supplying the receive clock.
2. TxC is TRxC or RTxC, whichever is supplying the transmit clock.
3. Both RTxC and SYNC have 30 pF capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between RxC and PCLK or TxC and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to chip PCLK requirements.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

System Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W}/\overline{REQ}$ Valid Delay	8	12	8	12	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	12	8	12	1,2
3	TdRXC(SY)	$\overline{RxC}$ ↑ to $\overline{SYNC}$ Valid Delay	4	7	4	7	2
4	TdRXC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	10	16	10	16	1,2
5	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W}/\overline{REQ}$ Valid Delay	5	8	5	8	3
6	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	8	5	8	1,3
7	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DTR}/\overline{REQ}$ Valid Delay	4	7	4	7	3
8	TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	6	10	6	10	1,3
9	TdSY(INT)	$\overline{SYNC}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	1
10	TdEXT(INT)	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	1

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.

* Timings are preliminary and subject to change.
† Units equal to TcPC.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8530	CE	4.0 MHz	SCC (40-pin)	Z8530A	CE	6.0 MHz	SCC (40-pin)
Z8530	CM	4.0 MHz	Same as above	Z8530A	CM	6.0 MHz	Same as above
Z8530	CMB	4.0 MHz	Same as above	Z8530A	CMB	6.0 MHz	Same as above
Z8530	CS	4.0 MHz	Same as above	Z8530A	CS	6.0 MHz	Same as above
Z8530	DE	4.0 MHz	Same as above	Z8530A	DE	6.0 MHz	Same as above
Z8530	DS	4.0 MHz	Same as above	Z8530A	DS	6.0 MHz	Same as above
Z8530	PE	4.0 MHz	Same as above	Z8530A	PE	6.0 MHz	Same as above
Z8530	PS	4.0 MHz	Same as above	Z8530A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to 125°C, MB = -55°C to 125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8531 ASCC
Asynchronous Serial Communications Controller

Zilog

Product Specification

September 1983

Features

- Two independent, 0 to 1M bit/second, full-duplex channels, each with a separate crystal oscillator and baud rate generator.
- Programmable for NRZ, NRZI, or FM data encoding.
- Local Loopback and Auto Echo modes.
- Asynchronous communications with five to eight bits per character and one, one and one-half, or two stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.

General Description

The Z8531 ASCC Asynchronous Serial Communications Controller is a dual-channel, multi-protocol data communications peripheral designed for use with conventional non-multiplexed buses. The ASCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators and crystal oscillators that dramatically reduce the need for external logic.

The ASCC also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

The Z-BUS daisy-chain interrupt hierarchy is also supported—as is standard for Zilog peripheral components.

The Z8531 ASCC is packaged in a 40-pin ceramic DIP and uses a single +5 V power supply.

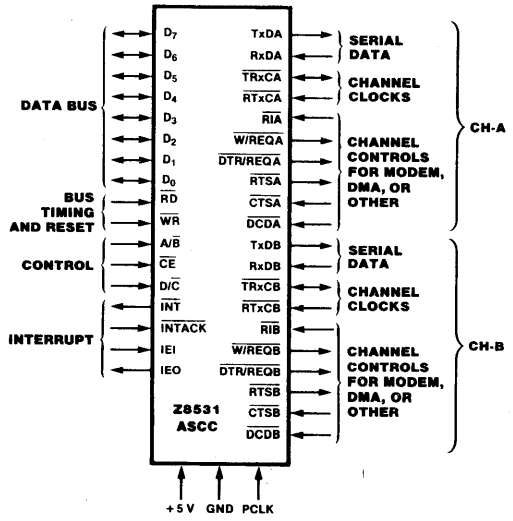


Figure 1. Pin Functions

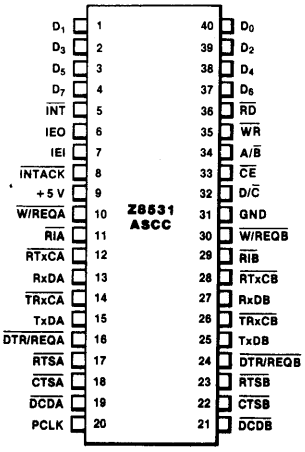


Figure 2. Pin Assignments

Pin Description

The following section describes the pin functions of the ASCC. Figures 1 and 2 detail the respective pin functions and pin assignments.

A/$\overline{B}$. *Channel A/Channel B Select* (input). This signal selects the channel in which the read or write operation occurs.

$\overline{CE}$. *Chip Enable* (input, active Low). This signal selects the ASCC for a read or write operation.

$\overline{CTSA}$, $\overline{CTSB}$. *Clear To Send* (inputs, active Low). If these pins are programmed as Auto Enables, a Low on the inputs enables the respective transmitters. If not programmed as Auto Enables, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The ASCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

D/$\overline{C}$. *Data/Control Select* (input). This signal defines the type of information transferred to or from the ASCC. A High means data is transferred; a Low indicates a command.

$\overline{DCDA}$, $\overline{DCDB}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accomodate slow rise-time signals. The ASCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

D_0-D_7. *Data Bus* (bidirectional, 3-state). These lines carry data and commands to and from the ASCC.

$\overline{DTR}$/$\overline{REQA}$, $\overline{DTR}$/$\overline{REQB}$. *Data Terminal Ready/Request* (outputs, active Low). These outputs follow the state programmed into the DTR bit. They can also be used as general-purpose outputs or as Request lines for a DMA controller.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an ASCC interrupt or the ASCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is activated when the ASCC requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This signal indicates an active Interrupt Acknowledge cycle. During this cycle, the ASCC interrupt daisy chain settles. When $\overline{RD}$ becomes active, the ASCC places an interrupt vector on the data bus (if IEI is High). $\overline{INTACK}$ is latched by the rising edge of PCLK.

PCLK. *Clock* (input). This is the master ASCC clock used to synchronize internal signals; PCLK is a TTL level signal.

$\overline{RD}$. *Read* (input, active Low). This signal indicates a read operation and when the ASCC is selected, enables the ASCC's bus drivers. During the Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the bus if the ASCC is the highest priority device requesting an interrupt.

RxDA, RxDB. *Receive Data* (inputs, active High). These input signals receive serial data at standard TTL levels.

$\overline{RIA}$, $\overline{RIB}$. *Ring Indicator* (inputs, active Low). These pins can act either as inputs, or part of the crystal oscillator circuit. In normal mode (crystal oscillator option not selected), these pins are inputs similar to $\overline{CTS}$ and $\overline{DCD}$. In this mode, transitions on these lines affect the state of the Ring Indicator status bits in Read Register 0 (Figure 8) but have no other function.

$\overline{RTxCA}$, $\overline{RTxCB}$. *Receive/Transmit Clocks* (inputs, active Low). These pins can be programmed in several different modes of operation. In each channel, $\overline{RTxC}$ may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective $\overline{RI}$ pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in Asynchronous modes.

$\overline{RTSA}$, $\overline{RTSB}$. *Request To Send* (outputs, active Low). When the Request To Send (RTS) bit in Write Register 5 (Figure 9) is set, the $\overline{RTS}$ signal goes Low. When the RTS bit is reset in the Asynchronous mode and Auto Enable is on, the signal goes High after the transmitter is empty. With Auto Enable off, the $\overline{RTS}$ pin strictly follows the state of the $\overline{RTS}$ bit. Both pins can be used as general-purpose outputs.

TxDA, TxDB. *Transmit Data* (outputs, active High). These output signals transmit serial data at standard TTL levels.

$\overline{TRxCA}$, $\overline{TRxCB}$. *Transmit/Receive Clocks* (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. $\overline{TRxC}$ may supply the receive clock or the transmit clock in the input mode or sup-

Pin Description (Continued)

ply the output of the Digital Phase-Locked Loop, the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

$\overline{WR}$. *Write* (input, active Low). When the ASCC is selected, this signal indicates a write operation. The coincidence of $\overline{RD}$ and $\overline{WR}$ is interpreted as a reset.

$\overline{W/REQA}$, $\overline{W/REQB}$. *Wait/Request* (outputs, open-drain when programmed for a Wait function, driven High or Low when programmed for a Request function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the ASCC data rate. The reset state is Wait.

Functional Description

The functional capabilities of the ASCC can be described from two different points of view: as a data communications device, it transmits and receives data in a wide variety of data communications protocols; as a microprocessor peripheral, the ASCC offers valuable features such as vectored interrupts, polling, and simple handshake capability.

Data Communications Capabilities. The ASCC provides two independent full-duplex channels programmable for use in any common Asynchronous data communication protocol. Figure 3 and the following description briefly detail this protocol.

Asynchronous Modes. Transmission and reception can be accomplished independently on each channel with five to eight bits per character, plus optional even or odd parity. The transmitters can supply one, one-and-a-half, or two stop bits per character and can provide a break output at any time. The receiver break-detection logic interrupts the CPU both at the start and at the end of a received break. Reception is protected from spikes by a transient spike-rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the receive data input (RxDA or RxDB in Figure 1). If the Low does not persist (as in the case of a transient), the character assembly process does not start.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occur. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids the interpretation of a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit begins.

The ASCC does not require symmetric transmit and receive clock signals—a feature allowing use of the wide variety of clock sources. The transmitter and receiver can handle data at a rate of 1/16, 1/32, or 1/64 of the clock rate supplied to the receive and transmit clock inputs.

Baud Rate Generator. Each channel in the ASCC contains a programmable baud rate generator. Each generator consists of two 8-bit time constant registers that form a 16-bit time constant, a 16-bit down counter, and a flip-flop on the output producing a square wave. On startup, the flip-flop on the output is set in a High state, the value in the time constant register is loaded into the counter, and the counter starts counting down. The output of the baud rate generator toggles upon reaching 0, the value in the time constant register is loaded into the counter, and the process is repeated. The time constant may be changed at any time, but the new value does not take effect until the next load of the counter.

The output of the baud rate generator may be used as either the transmit clock, the receive clock, or both. It can also drive the Digital Phase-Locked Loop (see next section).

If the receive clock or transmit clock is not programmed to come from the $\overline{TRxC}$ pin, the output of the baud rate generator may be echoed out via the $\overline{TRxC}$ pin.

The following formula relates the time constant to the baud rate (the baud rate is in bits/second and the BR clock period is in seconds):

$$\text{time constant} = \frac{PCLK}{2\ (\text{clock factor})\ (\text{baud})} - 2$$

Digital Phase-Locked Loop. The ASCC contains a Digital Phase-Locked-Loop (DPLL) to recover clock information from a data stream with NRZI or FM encoding. The DPLL is driven by a clock that is nominally 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a clock for the data. This clock may then be used as the ASCC receive clock, the transmit clock, or both.

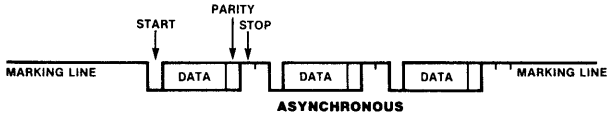


Figure 3. ASCC Protocol

Functional Description (Continued)

For NRZI encoding, the DPLL counts the 32x clock to create nominal bit times. As the 32x clock is counted, the DPLL is searching the incoming data stream for edges (either 1 to 0 or 0 to 1). Whenever an edge is detected, the DPLL makes a count adjustment (during the next counting cycle), producing a terminal count closer to the center of the bit cell.

For FM encoding, the DPLL still counts from 0 to 31, but with a cycle corresponding to two bit times. When the DPLL is locked, the clock edges in the data stream should occur between counts 15 and 16 and between counts 31 and 0. The DPLL looks for edges only during a time centered on the 15 to 16 counting transition.

The 32x clock for the DPLL can be programmed to come from either the $\overline{\text{RTxC}}$ input or the output of the baud rate generator. The DPLL output may be programmed to be echoed out of the ASCC via the $\overline{\text{TRxC}}$ pin (if this pin is not being used as an input).

Data Encoding. The ASCC may be programmed to encode and decode the serial data in four different ways (Figure 4). In NRZ encoding, a 1 is represented by a High level and a 0 is represented by a Low level. In NRZI encoding, a 1 is represented by no change in level and a 0 is represented by a change in level. In FM1 (more properly, bi-phase mark), a transition occurs at the beginning of every bit cell. A 1 is represented by an additional transition at the center of the bit cell and a 0 is represented by no additional transition at the center of the bit cell. In FM0 (bi-phase space), a transition occurs at the beginning of every bit cell. A 0 is represented by an additional transition at the center of the bit cell, and a 1 is represented by no additional transition at the center of the bit cell. In addition to these four methods, the ASCC can be used to decode Manchester (bi-phase level) data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoding always produces a transition at the center of the bit cell. If the transition is 0 to 1, the bit is a 0. If the transition is 1 to 0, the bit is a 1.

Auto Echo and Local Loopback. The ASCC is capable of automatically echoing everything it receives. In Auto Echo mode, RxD is connected to TxD internally. Auto Echo mode can be used with NRZI or FM encoding with no additional delay, because the data stream is not decoded before retransmission. In Auto Echo mode, the $\overline{\text{CTS}}$ input is ignored as a transmitter enable (although transitions on this input can still cause interrupts if programmed to do so). In this mode, the transmitter is actually bypassed and the programmer is responsible for disabling transmitter interrupts and $\overline{\text{WAIT/REQUEST}}$ on transmit.

The ASCC is also capable of local loopback. In this mode TxD is connected to RxD internally, just as in Auto Echo mode. However, in Local Loopback mode, the internal transmit data is tied to the internal receive data and RxD is ignored (except to be echoed out via TxD). The $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$ inputs are also ignored as transmit and receive enables. However, transitions on these inputs can still cause interrupts. Local Loopback works with NRZ, NRZI or FM coding of the data stream.

I/O Interface Capabilities. The ASCC offers the choice of Polling, Interrupt (vectored or nonvectored), and Block Transfer modes to transfer data, status, and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. All interrupts are disabled. Three status registers in the ASCC are automatically updated whenever any function is performed. The idea behind polling is for the CPU to periodically read a status register until the register contents indicate the need for data to be transferred. Only one register needs to be read; depending on its contents, the CPU either writes data, reads data, or continues. Two bits in the register indicate the need for

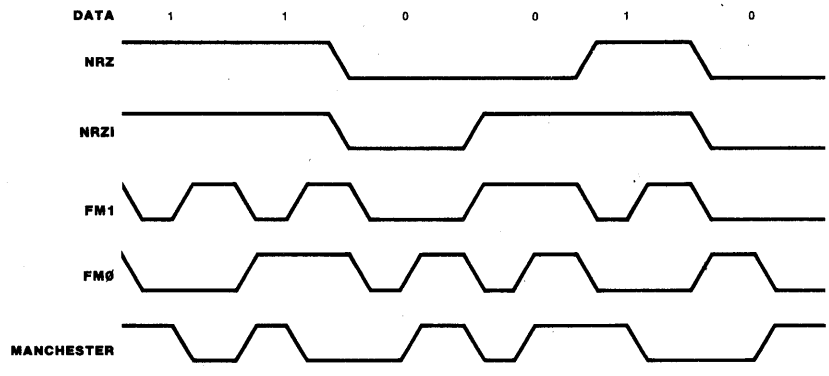


Figure 4. Data Encoding Methods

Functional Description (Continued)

data transfer. An alternative is a poll of the Interrupt Pending register to determine the source of an interrupt. The status for both channels resides in one register.

Interrupts. When an ASCC responds to an Interrupt Acknowledge signal ($\overline{INTACK}$) from the CPU, an interrupt vector may be placed on the data bus. This vector is written in WR2 and may be read in RR2A or RR2B (Figures 8 and 9).

To speed interrupt response time, the ASCC can modify three bits in this vector to indicate status. If the vector is read in Channel A, status is never included; if it is read in Channel B, status is always included.

Each of the six sources of interrupts in the ASCC (Transmit, Receive, and External/Status interrupts in both channels) has three bits associated with the interrupt source: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE). Operation of the IE bit is straightforward. If the IE bit is set for a given interrupt source, then that source can request interrupts. The exception is when the MIE (Master Interrupt Enable) bit in WR9 is reset and no interrupts may be requested. The IE bits are write only.

The other two bits are related to the interrupt priority chain (Figure 5). As a microprocessor peripheral, the ASCC may request an interrupt only when no higher priority device is requesting one, e.g., when IEI is High. If the device in question requests an interrupt, it pulls down $\overline{INT}$. The CPU then responds with $\overline{INTACK}$, and the interrupting device places the vector on the data bus.

In the ASCC, the IP bit signals a need for interrupt servicing. When an IP bit is 1 and the IEI input is High, the $\overline{INT}$ output is pulled Low, requesting an interrupt. In the ASCC, if the IE bit is not set by enabling interrupts, then the IP for that source can never be set. The IP bits are readable in RR3A.

The IUS bits signal that an interrupt request is being serviced. If an IUS is set, all interrupt sources of lower priority in the ASCC and external to the ASCC are prevented from requesting interrupts. The internal interrupt sources are inhibited by the state of the internal daisy chain, while lower priority devices are inhibited by the IEO output of the ASCC being pulled Low and propagated to subsequent peripherals. An IUS bit is set during an Interrupt Acknowledge cycle if there are no higher priority devices requesting interrupts.

There are three types of interrupts: Transmit, Receive, and External/Status. Each interrupt type is enabled under program control with Channel A having higher priority than Channel B, and with Receiver, Transmit, and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted when the transmit buffer becomes empty. (This implies that the transmitter must have had a data character written into it so that it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on First Receive Character or Special Receive Condition.
- Interrupt on All Receive Characters or Special Receive Condition.
- Interrupt on Special Receive Condition Only.

Interrupt on First Character or Special Condition and Interrupt on Special Condition Only are typically used with the Block Transfer mode. A Special Receive Condition is a receiver overrun, and, optionally, a parity error. The Special Receive Condition interrupt is different from an ordinary receive character available interrupt only in the status placed in the vector during the Interrupt Acknowledge cycle. In Interrupt on First Receive Character, an interrupt can occur from Special Receive Conditions any time after the first receive character interrupt.

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$, and $\overline{RI}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition, or a zero count in the baud rate generator, or by the detection of a Break.

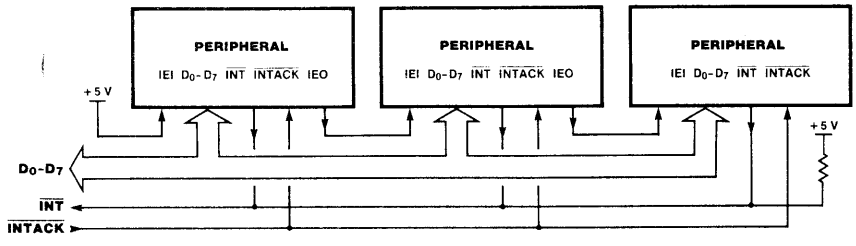


Figure 5. Interrupt Schedule

Functional Description (Continued)

CPU/DMA Block Transfer. The ASCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the $\overline{\text{WAIT}}$/ $\overline{\text{REQUEST}}$ output in conjunction with the Wait/Request bits in WR1. The $\overline{\text{WAIT}}$/ $\overline{\text{REQUEST}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{REQUEST}}$ line in the DMA Block Transfer mode.

To a DMA controller, the ASCC $\overline{\text{REQUEST}}$ output indicates that the ASCC is ready to transfer data to or from memory. To the CPU, the $\overline{\text{WAIT}}$ line indicates that the ASCC is not ready to transfer data, thereby requesting that the CPU extend the I/O cycle. The $\overline{\text{DTR}}$/ $\overline{\text{REQUEST}}$ line allows full-duplex operation under DMA control.

Architecture

The ASCC internal structure includes two full-duplex channels, two baud rate generators, internal control and interrupt logic, and a bus interface to a nonmultiplexed bus. Associated with each channel are a number of read and write registers for mode control and status information, as well as logic necessary to interface to modems or other external devices (Figure 6).

The logic for both channels provides formats, synchronization, and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general-purpose in nature and can optionally be used for functions other than modem control.

The register set for each channel includes ten control (write) registers, and four status (read) registers. In addition, each baud rate generator has two (read/write) registers for holding the time constant that determines the

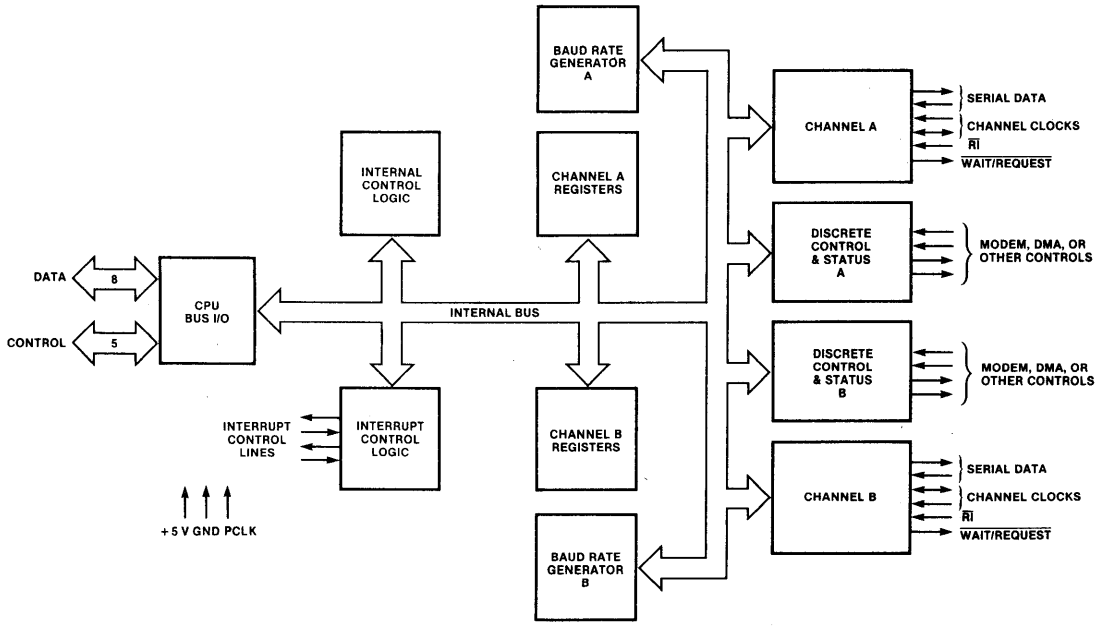


Figure 6. Block Diagram of ASCC Architecture

Architecture
(Continued)

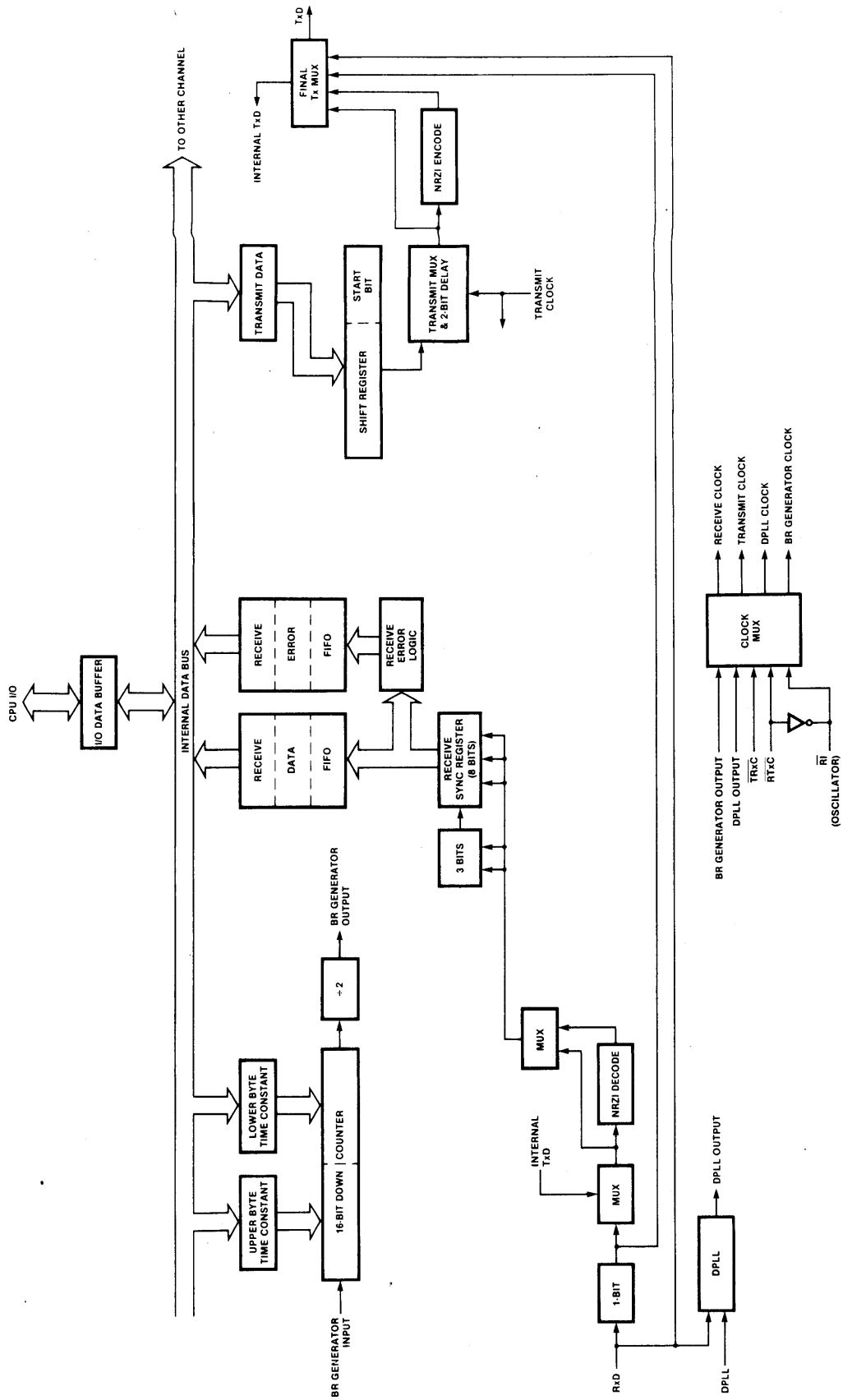


Figure 7. Data Path

Architecture (Continued)

baud rate. Finally, associated with the interrupt logic is a write register for the interrupt vector accessible through either channel, a write only Master Interrupt Control register and three read registers: one containing the vector with status infomation (Channel B only), one containing the vector without status (Channel A only), and one containing the Interrupt Pending bits (Channel A only).

The registers for each channel are designated as follows:

WR0–WR15 — Write Registers 0-5, 8-15.

RR0–RR3, RR10, RR12, RR13, RR15 — Read Registers 0 through 3, 10, 12, 13, 15.

Table 1 lists the functions assigned to each read or write register. The ASCC contains only one WR2 and WR9, but they can be accessed by either channel. All other registers are paired (one for each channel).

Data Path. The transmit and receive data path illustrated in Figure 7 is identical for both channels. The receiver has three 8-bit buffer registers in an FIFO arrangement, in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high speed data. Incoming data is routed through one of several paths depending on the selected mode (the character length also determines the data path).

The transmitter has an 8-bit Transmit Data buffer register loaded from the internal data bus and an 11-bit Transmit Shift register that can be loaded from the Transmit Data register.

Read Register Functions	
RR0	Transmit/Receive buffer status and External status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only) Unmodified interrupt vector (Channel A only)
RR3	Interrupt Pending bits (Channel A only)
RR8	Receive buffer
RR10	Miscellaneous status
RR12	Lower byte of baud rate generator time constant
RR13	*Upper byte of baud rate generator time constant*
RR15	External/Status interrupt information

Write Register Functions	
WR0	CRC initialize, initialization commands for the various modes, Register Pointers.
WR1	Transmit/Receive interrupt and data transfer mode definition
WR2	Interrupt vector (accessed through either channel)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR8	Transmit buffer
WR9	Master interrupt control and reset (accessed through either channel)
WR10	Miscellaneous transmitter/receiver control bits
WR11	Clock mode control
WR12	Lower byte of baud rate generator time constant
WR13	Upper byte of baud rate generator time constant
WR14	Miscellaneous control bits
WR15	External/Status interrupt control

Table 1. Read and Write Register Functions

Programming

The ASCC contains 11 write registers in each channel that are programmed by the system separately to configure the functional personality of the channels.

In the ASCC, register addressing is direct for the data registers only, which are selected by a High on the D/$\overline{C}$ pin. In all other cases (with the exception of WR0 and RR0), programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to WR0 and contains three bits that point to the selected register. The second write is the actual control word for the selected register, and if the second operation is read, the selected read register is accessed. All of the registers in the ASCC, including the data registers, may be accessed in this fashion. The pointer bits are automatically cleared after the read or write operation so that WR0 (or RR0) is addressed again.

The system program first issues a series of commands to initialize the basic mode of operation. For example, the character length, clock rate, number of stop bits, even or odd parity might be set first. Then the interrupt mode would be set, and finally, receiver or transmitter enable.

Programming (Continued)

Read Registers. The ASCC contains eight read registers (actually nine, counting the receive buffer (RR8) in each channel). Four of these may be read to obtain status information (RR0, RR1, RR10, and RR15). Two registers (RR12 and RR13) may be read to learn the baud rate generator time constant. RR2 contains either the unmodified interrupt vector (Channel A) or the vector modified by status information (Channel B). RR3 contains the Interrupt Pending (IP) bits (Channel A). Figure 8 shows the formats for each read register.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring; e.g., when the interrupt vector indicates a Special Receive Condition interrupt, all the appropriate error bits can be read from a single register (RR1).

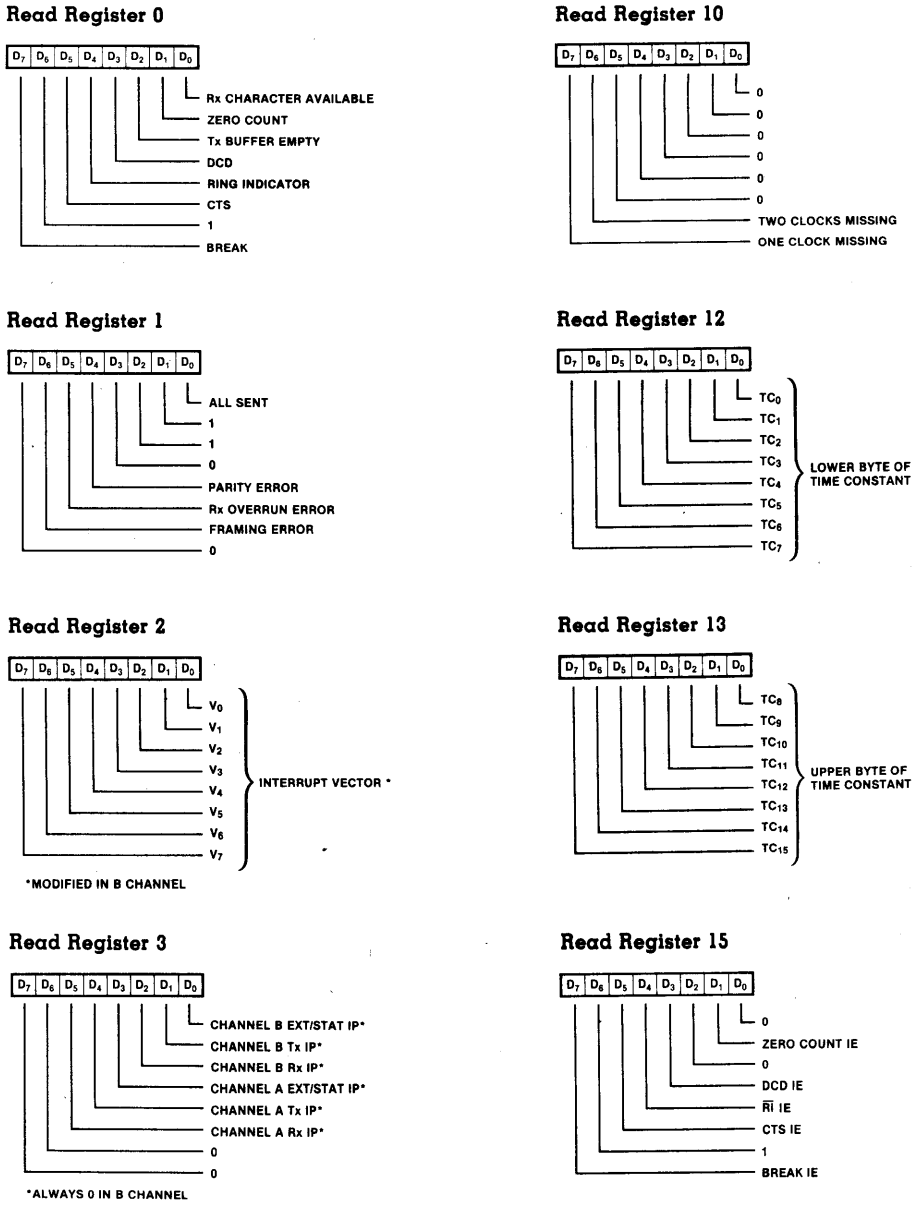
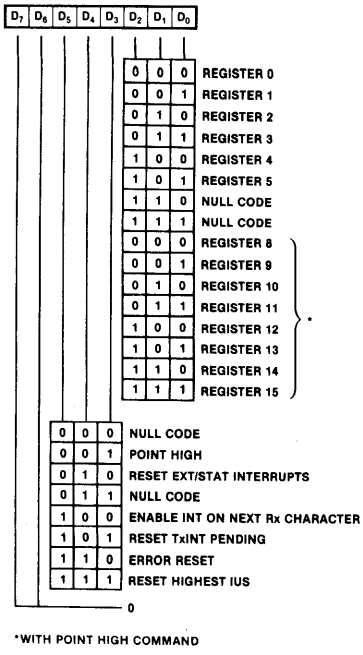


Figure 8. Read Register Bit Functions

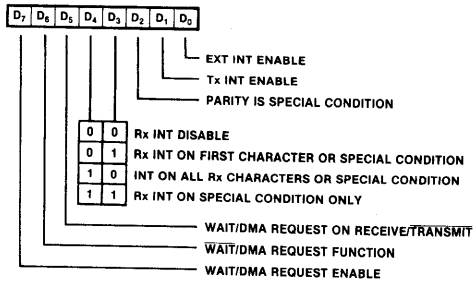
Programming (Continued)

Write Registers. The ASCC contains 11 write registers (12 counting WR8, the transmit buffer) in each channel. These write registers are programmed separately to configure the functional "personality" of the channels. In addition, there are two registers (WR2 and WR9) shared by the two channels that may be accessed through either of them. WR2 contains the interrupt vector for both channels, while WR9 contains the interrupt control bits. Figure 9 shows the format of each write register.

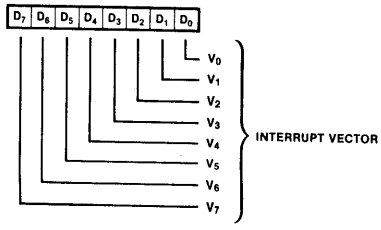
Write Register 0



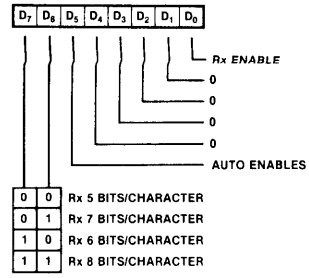
Write Register 1



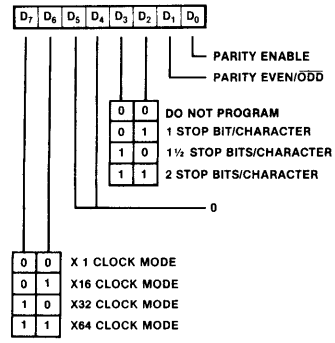
Write Register 2



Write Register 3



Write Register 4



Write Register 5

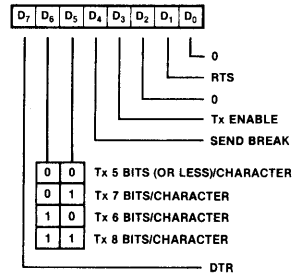
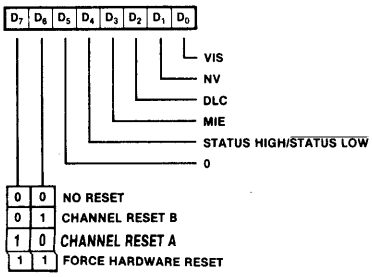


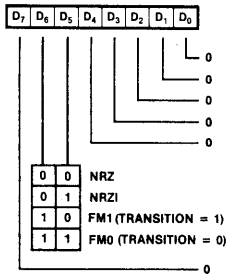
Figure 9. Write Register Bit Functions

Programming (Continued)

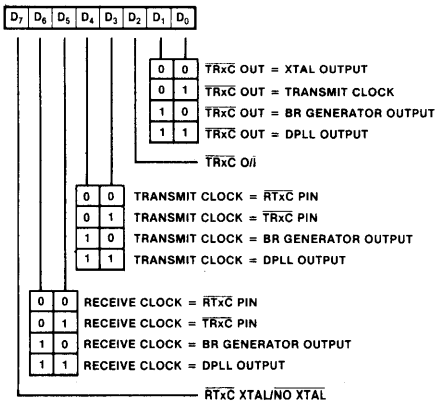
Write Register 9



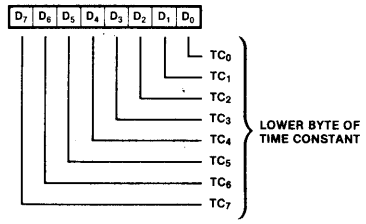
Write Register 10



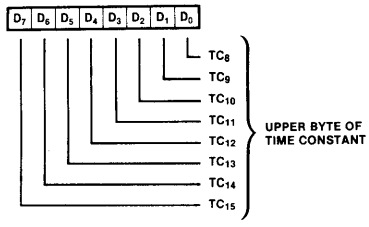
Write Register 11



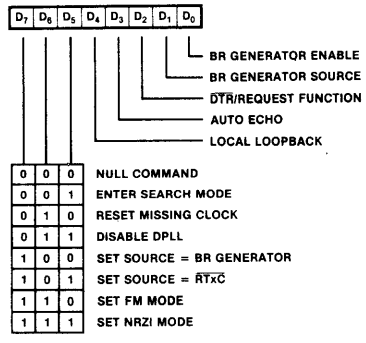
Write Register 12



Write Register 13



Write Register 14



Write Register 15

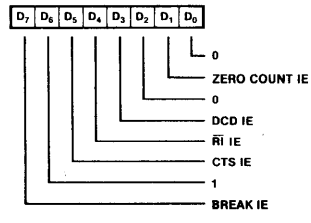


Figure 9. Write Register Bit Functions (Continued)

Timing

The ASCC generates internal control signals from $\overline{WR}$ and $\overline{RD}$ that are related to PCLK. Since PLCK has no phase relationship with $\overline{WR}$ and $\overline{RD}$, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK. The recovery time applies only between bus transactions involving the ASCC. The recovery time required for proper operation is specified from the rising edge of $\overline{WR}$ or $\overline{RD}$ in the first transaction involving the ASCC to the falling edge of $\overline{WR}$ or $\overline{RD}$ in the second transaction involving the ASCC. This time must be at least 6 PLCK cycles plus 200 ns.

Read Cycle Timing. Figure 10 illustrates read cycle timing, Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{RD}$ falls,or rises before $\overline{RD}$ rises, the effective $\overline{RD}$ is shortened.

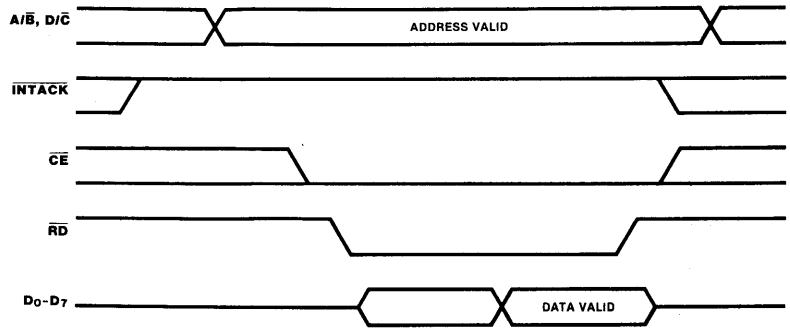


Figure 10. Read Cycle Timing

Write Cycle Timing. Figure 11 illustrates write cycle timing. Addresses on A/$\overline{B}$ and D/$\overline{C}$ and the status on $\overline{INTACK}$ must remain stable throughout the cycle. If $\overline{CE}$ falls after $\overline{WR}$ falls or rises before $\overline{WR}$ rises, the effective $\overline{WR}$ is shortened.

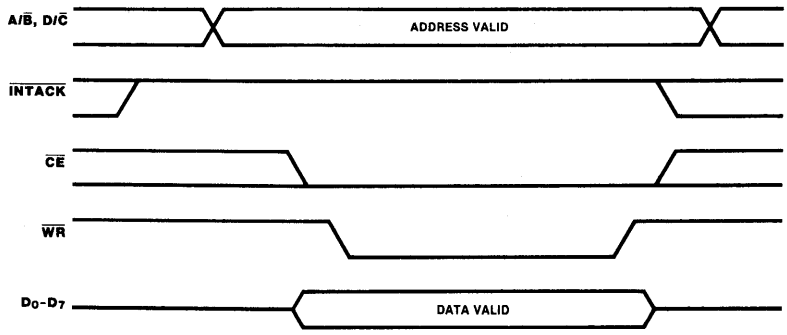


Figure 11. Write Cycle Timing

Interrupt Acknowledge Cycle Timing. Figure 12 illustrates interrupt acknowledge cycle timing. Between the time $\overline{INTACK}$ goes low and the falling edge of $\overline{RD}$, the internal and external IEI/IEO daisy chains settle. If there is an interrupt pending the ASCC and IEI is High when $\overline{RD}$ falls, the acknowledge cycle was intended for the ASCC. In this case, the ASCC may be programmed to respond to $\overline{RD}$ Low by placing its interrupt vector on D_0–D_7 and sets the appropriate Interrupt-Under-Service latch internally.

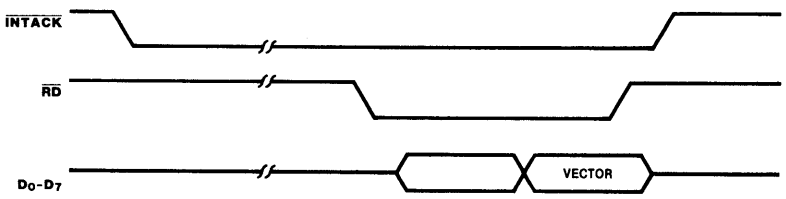


Figure 12. Interrupt Acknowledge Cycle Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

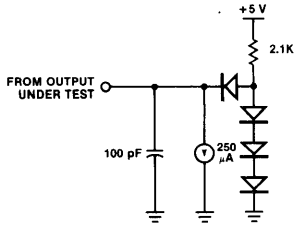


Figure 13. Standard Test Load

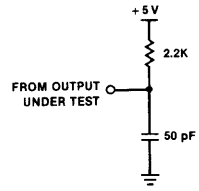


Figure 14. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0\ mA$
I_{IL}	Input Leakage		±10.0	µA	$0.4 \leq V_{IN} \leq +2.4V$
I_{OL}	Output Leakage		±10.0	µA	$0.4 \leq V_{OUT} \leq +2.4V$
I_{CC}	V_{CC} Supply Current		250	mA	

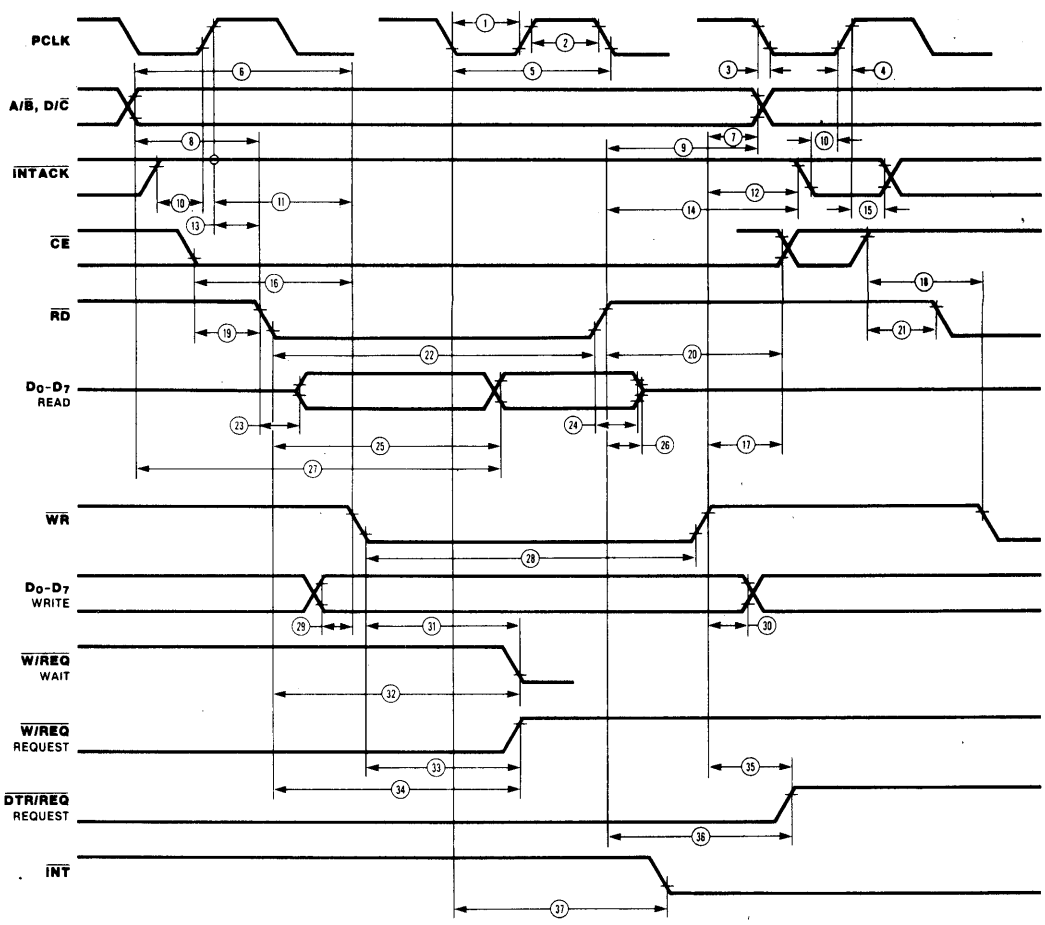
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

Read and Write Timing

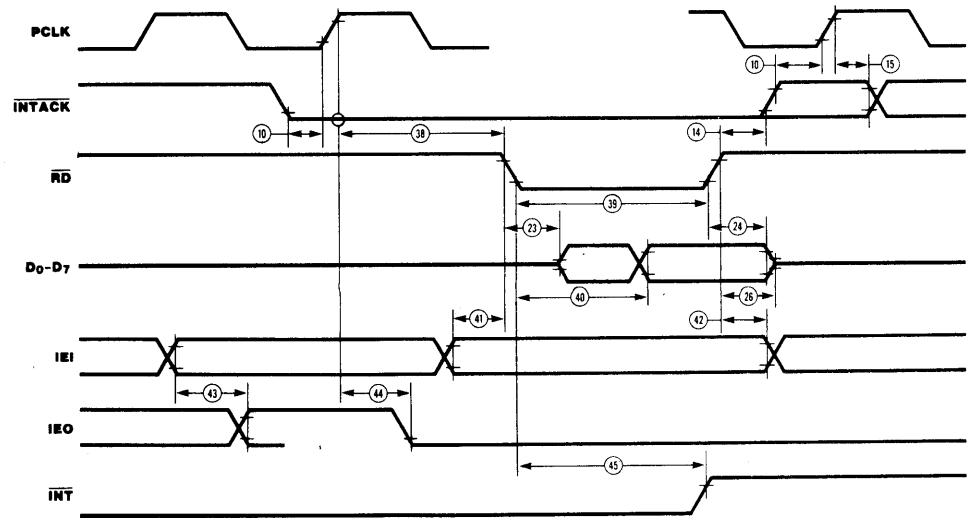


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TwPCl	PCLK Low Width	105	2000	70	1000	
2	TwPCh	PCLK High Width	105	2000	70	1000	
3	TfPC	PCLK Fall Time		20		10	
4	TrPC	PCLK Rise Time		20		15	
5	TcPC	PCLK Cycle Time	250	4000	165	2000	
6	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		80		
7	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		
8	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		80		
9	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		
10	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	0		0		
11	TsIAi(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		200		1
12	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		
13	TsIAi(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		200		1
14	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		
15	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	100		100		
16	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		
17	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		
18	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		70		
19	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		1
20	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		1
21	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		70		1
22	TwRDl	$\overline{RD}$ Low Width	390		250		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
24	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
25	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		250		180	
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70		45	2

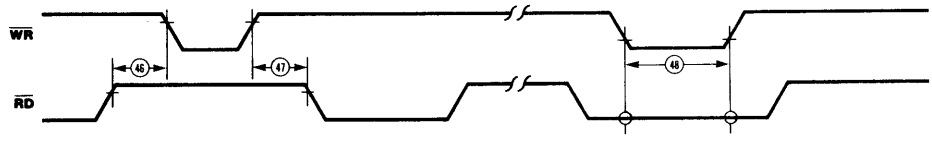
NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is defined as the time required for a ±0.5 V change in the output with a maximum dc load and minimum ac load.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

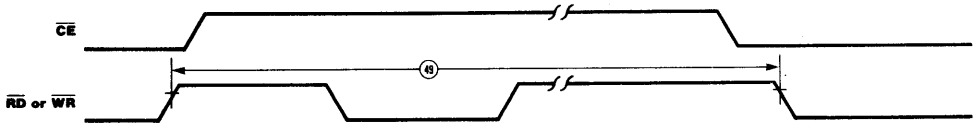
Interrupt Acknowledge Timing



Reset Timing



Cycle Timing



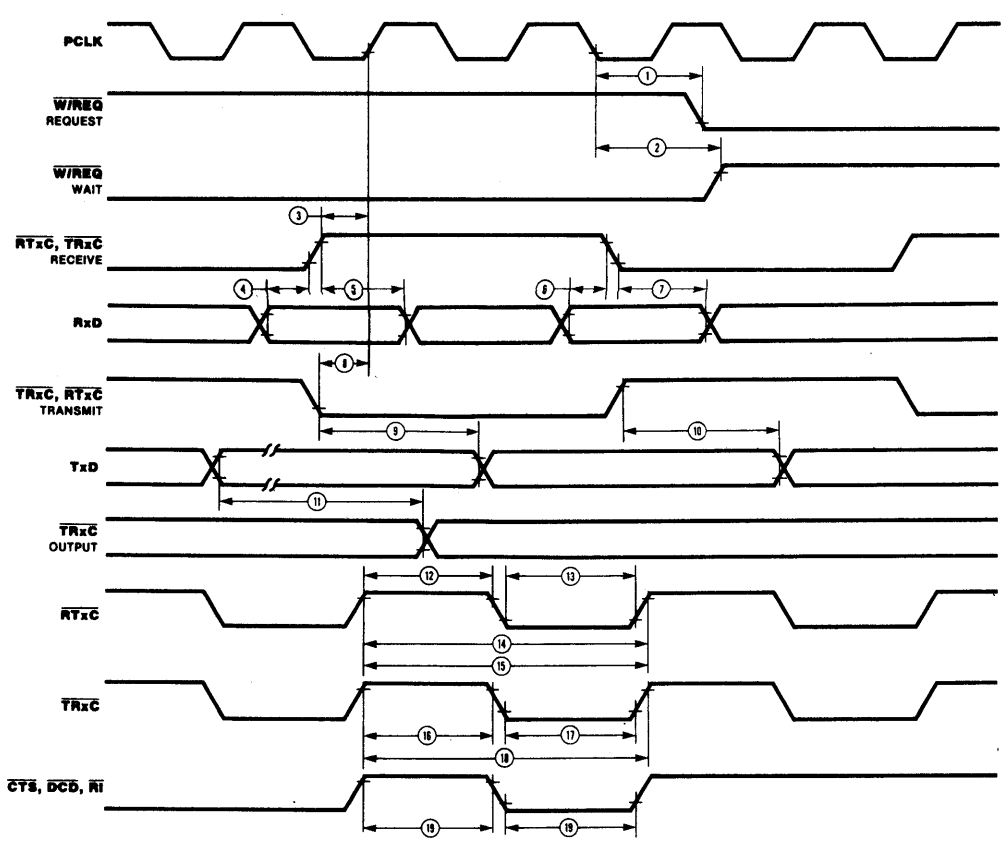
No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
27	TdA(DR)	Address Required Valid to Read Data Valid Delay		590		420	
28	TwWRl	WR Low Width	390		250		
29	TsDW(WR)	Write Data to WR ↓ Setup Time	0		0		
30	ThDW(WR)	Write Data to WR ↑ Hold Time	0		0		
31	TdWR(W)	WR ↓ to Wait Valid Delay		240		200	4
32	TdRD(W)	RD ↓ to Wait Valid Delay		240		200	4
33	TdWRf(REQ)	WR ↓ to W/REQ Not Valid Delay		240		200	
34	TdRDf(REQ)	RD ↓ to W/REQ Not Valid Delay		240		200	
35	TdWRr(REQ)	WR ↑ to DTR/REQ Not Valid Delay		5TcPC +300		5TcPC +250	
36	TdRDr(REQ)	RD ↑ to DTR/REQ Not Valid Delay		5TcPC +300		5TcPC +250	
37	TdPC(INT)	PCLK ↓ to INT Valid Delay		500		500	4
38	TdIAi(RD)	INTACK to RD ↓ (Acknowledge) Delay					5
39	TwRDA	RD (Acknowledge) Width	285		250		
40	TdRDA(DR)	RD ↓ (Acknowledge) to Read Data Valid Delay		190		180	
41	TsIEI(RDA)	IEI to RD ↓ (Acknowledge) Setup Time	120		100		
42	ThIEI(RDA)	IEI to RD ↑ (Acknowledge) Hold Time	0		0		
43	TdIEI(IEO)	IEI to IEO Delay Time		120		100	
44	TdPC(IEO)	PCLK ↑ to IEO Delay		250		250	
45	TdRDA(INT)	RD ↓ to INT Inactive Delay		500		500	4
46	TdRD(WRQ)	RD ↑ to WR ↓ Delay for No Reset	30		15		
47	TdWRQ(RD)	WR ↑ to RD ↓ Delay for No Reset	30		30		
48	TwRES	WR and RD Coincident Low for Reset	250		250		
49	Trc	Valid Access Recovery Time	6TcPC +200		6TcPC +130		3

NOTES:

3. Parameter applies only between transactions involving the ASCC.
4. Open-drain output, measured with open-drain test load.
5. Parameter is system dependent. For any ASCC in the daisy chain, TdIAi(RD) must be greater than the sum of TdPC(IEO) for the highest priority device in the daisy chain, TsIEI(RDA) for the ASCC, and TdIEI(IEO) for each device separating them in the daisy chain.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

General Timing

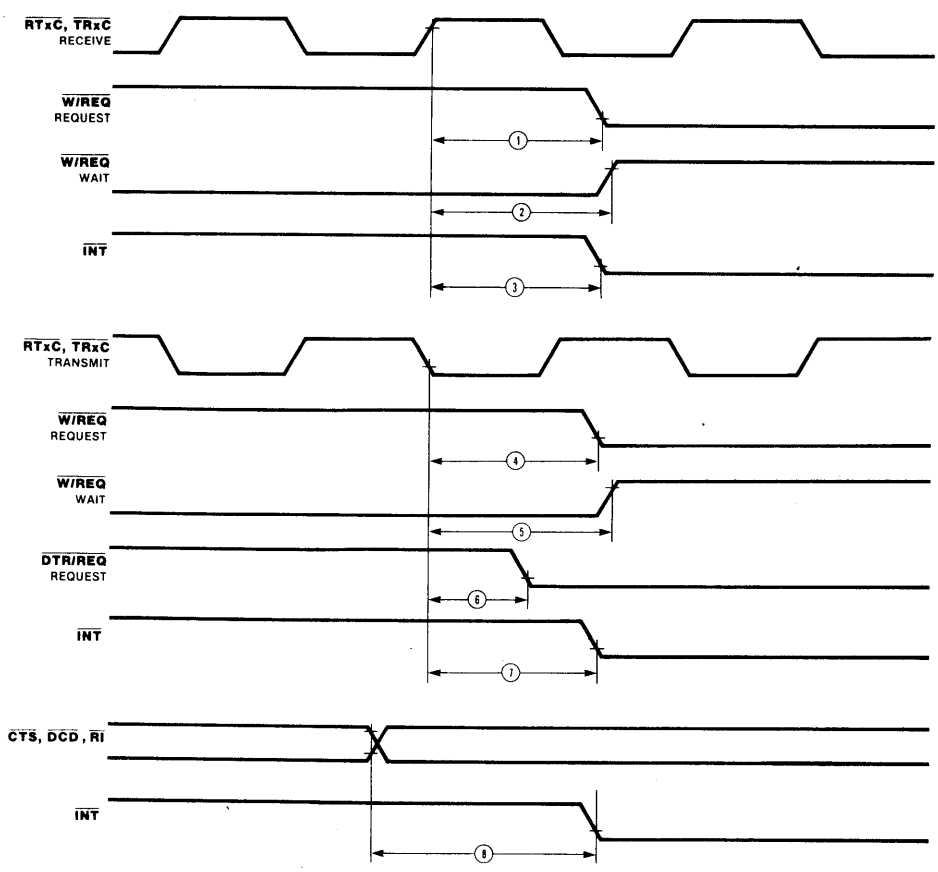


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdPC(REQ)	PCLK ↓ to W/$\overline{REQ}$ Valid Delay		250		250	
2	TdPC(W)	PCLK ↓ to Wait Inactive Delay		350		350	
3	TsRXC(PC)	$\overline{RxC}$ ↑ to PCLK ↑ Setup Time (PCLK ÷ 4 case only)	80	TwPCl	70	TwPCl	1,4
4	TsRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Setup Time (X1 Mode)	0		0		1
5	ThRXD(RXCr)	RxD to $\overline{RxC}$ ↑ Hold Time (X1 Mode)	150		150		1
6	TsRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Setup Time (X1 Mode)	0		0		1,5
7	ThRXD(RXCf)	RxD to $\overline{RxC}$ ↓ Hold Time (X1 Mode)	150		150		1,5
8	TsTXC(PC)	$\overline{TxC}$ ↓ to PCLK ↑ Setup Time	0		0		2,4
9	TdTXCf(TXD)	$\overline{TxC}$ ↓ to TxD Delay (X1 Mode)		300		300	2
10	TdTXCr(TXD)	$\overline{TxC}$ ↑ to TxD Delay (X1 Mode)		300		300	2,5
11	TdTXD(TRX)	TxD to $\overline{TRxC}$ Delay (Send Clock Echo)					
12	TwRTXh	$\overline{RTxC}$ High Width	180		180		6
13	TwRTXl	$\overline{RTxC}$ Low Width	180		180		6
14	TcRTX	$\overline{RTxC}$ Cycle Time	400		400		6
15	TcRTXX	Crystal Oscillator Period	250	1000	250	1000	3
16	TwTRXh	$\overline{TRxC}$ High Width	180		180		6
17	TwTRXl	$\overline{TRxC}$ Low Width	180		180		6
18	TcTRX	$\overline{TRxC}$ Cycle Time	400		400		6
19	TwEXT	$\overline{DCD}$ or $\overline{CTS}$ or RI Pulse Width	200		200		

NOTES:
1. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
2. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.
3. Both $\overline{RTxC}$ and $\overline{RI}$ have 30 pF capacitors to ground connected to them.
4. Parameter applies only if the data rate is one-fourth the PCLK rate. In all other cases, no phase relationship between $\overline{RxC}$ and PCLK or $\overline{TxC}$ and PCLK is required.
5. Parameter applies only to FM encoding/decoding.
6. Parameter applies only for transmitter and receiver; DPLL and baud rate generator timing requirements are identical to chip PCLK requirements.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).

System Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRXC(REQ)	$\overline{RxC}$ ↑ to $\overline{W/REQ}$ Valid Delay	8	12	8	12	2
2	TdRXC(W)	$\overline{RxC}$ ↑ to Wait Inactive Delay	8	12	8	12	1,2
3	TdRXC(INT)	$\overline{RxC}$ ↑ to $\overline{INT}$ Valid Delay	10	16	10	16	1,2
4	TdTXC(REQ)	$\overline{TxC}$ ↓ to $\overline{W/REQ}$ Valid Delay	5	8	5	8	3
5	TdTXC(W)	$\overline{TxC}$ ↓ to Wait Inactive Delay	5	8	5	8	1,3
6	TdTXC(DRQ)	$\overline{TxC}$ ↓ to $\overline{DTR/REQ}$ Valid Delay	4	7	4	7	3
7	TdTXC(INT)	$\overline{TxC}$ ↓ to $\overline{INT}$ Valid Delay	6	10	6	10	1,3
8	TdEXT(INT)	$\overline{DCD}$ or $\overline{CTS}$ Transition to $\overline{INT}$ Valid Delay	2	6	2	6	1

NOTES:
1. Open-drain output, measured with open-drain test load.
2. $\overline{RxC}$ is $\overline{RTxC}$ or $\overline{TRxC}$, whichever is supplying the receive clock.
3. $\overline{TxC}$ is $\overline{TRxC}$ or $\overline{RTxC}$, whichever is supplying the transmit clock.

* Timings are preliminary and subject to change.
† Units equal to TcPC.

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8531	CE	4.0 MHz	ASCC (40-pin)	Z8531A	CE	6.0 MHz	ASCC (40-pin)
Z8531	CM	4.0 MHz	Same as above	Z8531A	CM	6.0 MHz	Same as above
Z8531	CMB	4.0 MHz	Same as above	Z8531A	CMB	6.0 MHz	Same as above
Z8531	CS	4.0 MHz	Same as above	Z8531A	CS	6.0 MHz	Same as above
Z8531	DE	4.0 MHz	Same as above	Z8531A	DE	6.0 MHz	Same as above
Z8531	DS	4.0 MHz	Same as above	Z8531A	DS	6.0 MHz	Same as above
Z8531	PE	4.0 MHz	Same as above	Z8531A	PE	6.0 MHz	Same as above
Z8531	PS	4.0 MHz	Same as above	Z8531A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; CM = -55°C to +125°C, E = -40°C to +85°C, M = -55°C to 125°C, MB = -55°C to +125°C with MIL-STD-883 Class B processing, S = 0°C to +70°C.

Z8536 CIO Counter/Timer and Parallel I/O Unit

Zilog

Product Specification

September 1983

Features

- Two independent 8-bit, double-buffered, bidirectional I/O ports plus a 4-bit special-purpose I/O port. I/O ports feature programmable polarity, programmable direction (Bit mode), "pulse catchers," and programmable open-drain outputs.
- Four handshake modes, including 3-Wire (like the IEEE-488).
- REQUEST/$\overline{\text{WAIT}}$ signal for high-speed data transfer.
- Flexible pattern-recognition logic, programmable as a 16-vector interrupt controller.
- Three independent 16-bit counter/timers with up to four external access lines per counter/timer (count input, output, gate, and trigger), and three output duty cycles (pulsed, one-shot, and square-wave), programmable as retriggerable or nonretriggerable.
- Easy to use since all registers are read/write.

General Description

The Z8536 CIO Counter/Timer and Parallel I/O element is a general-purpose peripheral circuit, satisfying most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. In addition, each register is given its own unique internal address, so that any register can be accessed in two operations. All data registers can be directly accessed in a single operation. The CIO is easily interfaced to all popular microprocessors.

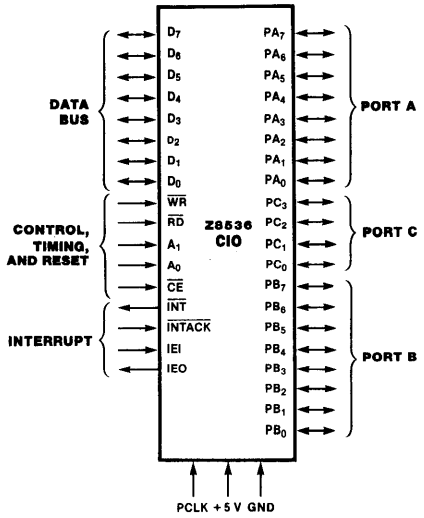


Figure 1. Pin Functions

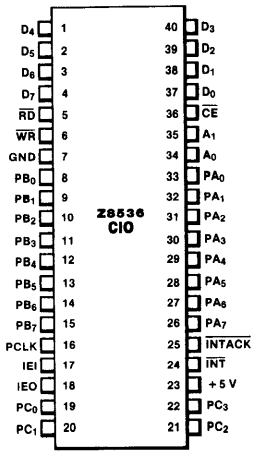


Figure 2. Pin Assignments

Pin Description

A_0-A_1. *Address Lines* (input). These two lines are used to select the register involved in the CPU transaction: Port A's Data register, Port B's Data register, Port C's Data register, or a control register.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level on this input enables the CIO to be read from or written to.

D_0-D_7. *Data Bus* (bidirectional 3-state). These eight data lines are used for transfers between the CPU and the CIO.

IEI. *Interrupt Enable In* (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device. A High IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from the requesting CIO or is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

$\overline{INT}$. *Interrupt Request* (output, open-drain, active Low). This signal is pulled Low when the CIO requests an interrupt.

$\overline{INTACK}$. *Interrupt Acknowledge* (input, active Low). This input indicates to the CIO that an Interrupt Acknowledge cycle is in progress. $\overline{INTACK}$ must be synchronized to PCLK, and it must be stable throughout the Interrupt Acknowledge cycle.

PA_0-PA_7. *Port A I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port A and external devices.

PB_0-PB_7. *Port B I/O lines* (bidirectional, 3-state, or open-drain). These eight I/O lines transfer information between the CIO's Port B and external devices. May also be used to provide external access to Counter/Timers 1 and 2.

PC_0-PC_3. *Port C I/O lines* (bidirectional, 3-state, or open-drain). These four I/O lines are used to provide handshake, $\overline{WAIT}$, and REQUEST lines for Ports A and B or to provide external access to Counter/Timer 3 or access to the CIO's Port C.

PCLK. *Peripheral Clock* (input, TTL-compatible). This is the clock used by the internal control logic and the counter/timers in timer mode. It does not have to be the CPU clock.

$\overline{RD}$*. *Read* (input, active Low). This signal indicates that a CPU is reading from the CIO. During an Interrupt Acknowledge cycle, this signal gates the interrupt vector onto the data bus if the CIO is the highest priority device requesting an interrupt.

$\overline{WR}$*. *Write* (input, active Low). This signal indicates a CPU write to the CIO.

*When $\overline{RD}$ and $\overline{WR}$ are detected Low at the same time (normally an illegal condition), the CIO is reset.

Architecture

The CIO Counter/Timer and Parallel I/O element (Figure 3) consists of a CPU interface, three I/O ports (two general-purpose 8-bit ports and one special-purpose 4-bit port),

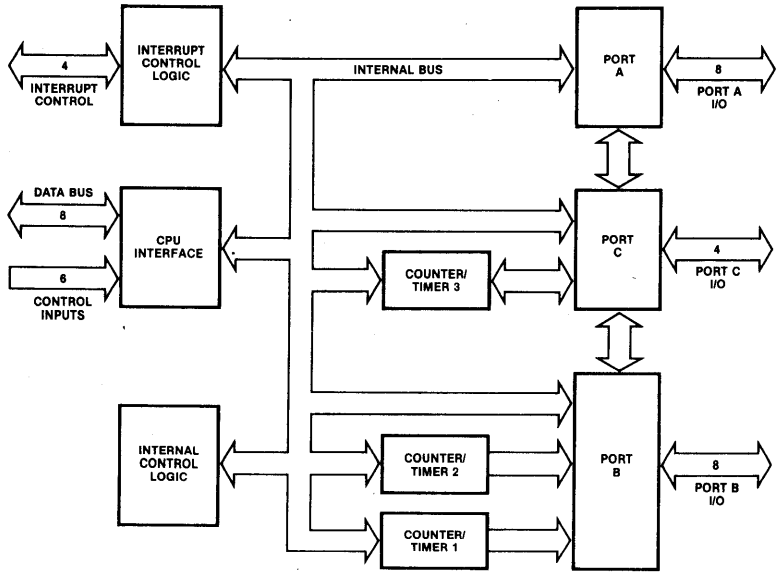


Figure 3. CIO Block Diagram

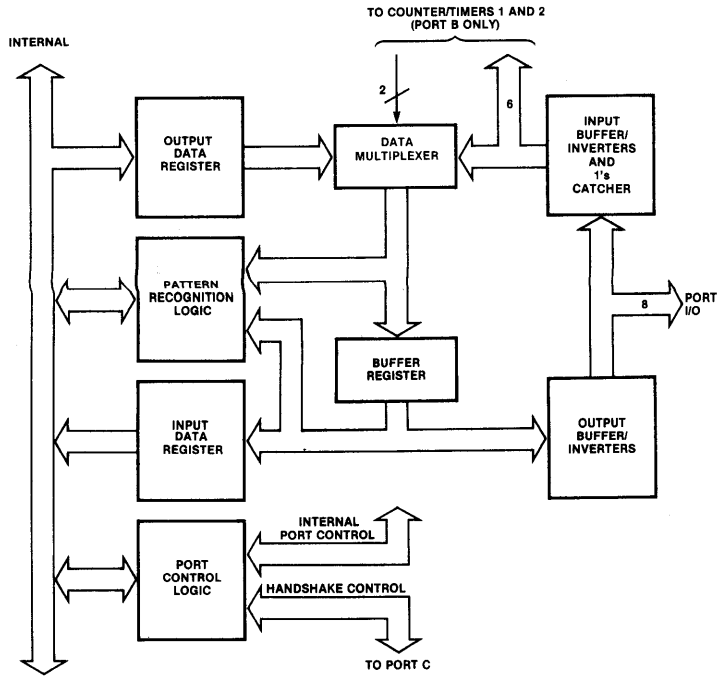


Figure 4. Ports A and B Block Diagram

three 16-bit counter/timers, an interrupt-control logic block, and the internal-control logic block. An extensive number of programmable options allow the user to tailor the configuration to best suit the specific application.

The two general-purpose 8-bit I/O ports (Figure 4) are identical, except that Port B can be specified to provide external access to Counter/Timers 1 and 2. Either port can be programmed to be a handshake-driven, double-buffered port (input, output, or bidirectional) or a control-type port with the direction of each bit individually programmable. Each port includes pattern-recognition logic, allowing interrupt generation when a specific pattern is detected. The pattern-recognition logic can be programmed so the port functions like a priority-interrupt controller. Ports A and B can also be linked to form a 16-bit I/O port.

To control these capabilities, both ports contain 12 registers. Three of these registers, the Input, Output, and Buffer registers, comprise the data path registers. Two registers, the Mode Specification and Handshake Specification registers, are used to define the mode of the port and to specify which handshake, if any, is to be used. The reference pattern for the pattern-recognition logic is defined via three registers: the Pattern Polarity, Pattern Transition, and Pattern Mask registers. The detailed characteristics of each bit path (for example, the direction of data flow or whether a path is inverting or noninverting) are programmed using the Data Path Polarity, Data Direction, and Special I/O Control registers.

The primary control and status bits are grouped in a single register, the Command and Status register, so that after the port is initially configured, only this register must be accessed frequently. To facilitate initialization, the port logic is designed so that registers associated with an unrequired capability are ignored and do not have to be programmed.

The function of the special-purpose 4-bit port, Port C (Figure 5), depends upon the roles of Ports A and B. Port C provides the required handshake lines. Any bits of Port C not used as handshake lines can be used as I/O lines or to provide external access for the third counter/timer.

Since Port C's function is defined primarily by Ports A and B, only three registers (besides the Data Input and Output registers) are needed. These registers specify the details of each bit path: the Data Path Polarity, Data Direction, and Special I/O Control registers.

Architecture
(Continued)

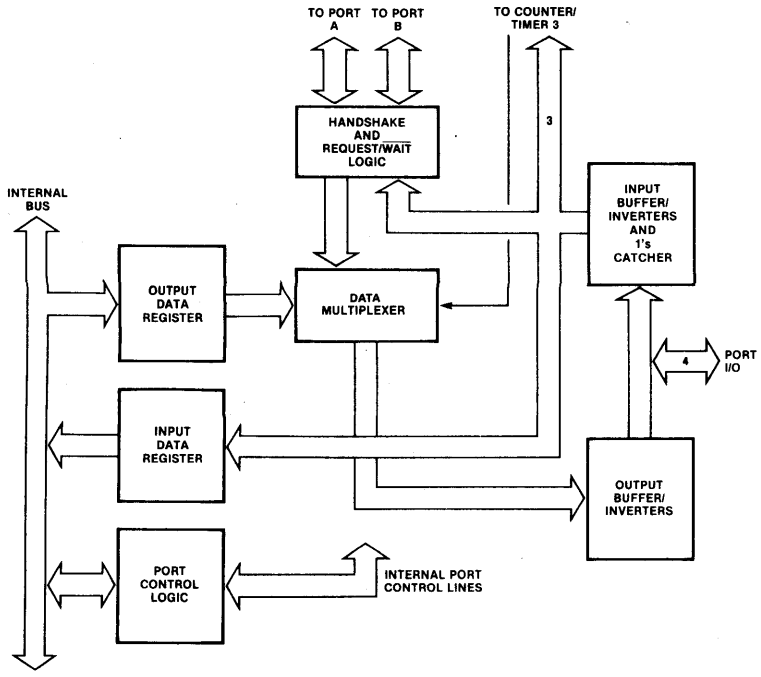


Figure 5. Port C Block Diagram

The three counter/timers (Figure 6) are all identical. Each is comprised of a 16-bit down-counter, a 16-bit Time Constant register (which holds the value loaded into the down-counter), a 16-bit Current Count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the Mode Specification and the Command and Status registers).

The capabilities of the counter/timer are numerous. Up to four port I/O lines can be dedicated as external access lines for each counter/timer: counter input, gate input, trigger input, and counter/timer output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square-wave. The operation of the counter/timer can be programmed as either retriggerable or nonretriggerable. With these and other options, most counter/timer applications are covered.

There are five registers (Master Interrupt Control register, three Interrupt Vector registers, and the Current Vector register) associated with the interrupt logic. In addition, the ports' Command and Status registers and the counter/timers' Command and Status registers include bits associated with the interrupt logic. Each of these registers contains three bits for interrupt control and status: Interrupt Pending (IP), Interrupt Under Service (IUS), and Interrupt Enable (IE).

Architecture
(Continued)

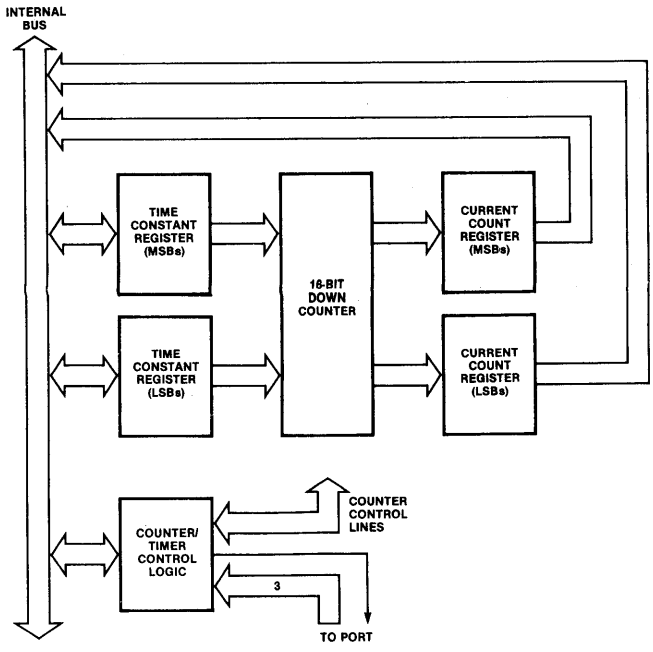


Figure 6. Counter/Timer Block Diagram

Functional Description

The following describes the functions of the ports, pattern-recognition logic, counter/timers, and interrupt logic.

I/O Port Operations. Of the CIO's three I/O ports, two (Ports A and B) are general-purpose, and the third (Port C) is a special-purpose 4-bit port. Ports A and B can be configured as input, output, or bidirectional ports with handshake. (Four different handshakes are available.) They can also be linked to form a single 16-bit port. If they are not used as ports with handshake, they provide 16 input or output bits with the data direction programmable on a bit-by-bit basis. Port B also provides access for Counter/Timers 1 and 2. In all configurations, Ports A and B can be programmed to recognize specific data patterns and to generate interrupts when the pattern is encountered.

The four bits of Port C provide the handshake lines for Ports A and B when required. A REQUEST/$\overline{\text{WAIT}}$ line can also be provided so that CIO transfers can be synchronized with DMAs or CPUs. Any Port C bits not used for handshake or REQUEST/$\overline{\text{WAIT}}$ can be used as input or output bits (individually data-direction programmable) or external access lines for Counter/Timer 3. Port C does not contain any pattern-recognition logic. It is, however, capable of bit-addressable writes. With this feature, any combination of bits can be set and/or cleared while the other bits remain undisturbed without first reading the register.

Bit Port Operations. In bit port operations, the port's Data Direction register specifies the direction of data flow for each bit. A 1 specifies an input bit, and a 0 specifies an output bit. If bits are used as I/O bits for a counter/timer, they should be set as input or output, as required.

The Data Path Polarity register provides the capability of inverting the data path. A 1 specifies inverting, and a 0 specifies noninverting. All discussions of the port operations assume that the path is noninverting.

The value returned when reading an input bit reflects the state of the input just prior to the read. A 1's catcher can be inserted into the input data path by programming a 1 to the corresponding bit position of the port's Special I/O Control register. When a 1 is detected at the 1's catcher input, its output is set to 1 until it is cleared. The 1's catcher is cleared by writing a 0 to the bit. In all other cases, attempted writes to input bits are ignored.

When Ports A and B include output bits, reading the Data register returns the value being output. Reads of Port C return the state of the pin. Outputs can be specified as open-drain by writing a 1 to the corresponding bit of the port's Special I/O Control register. Port C has the additional feature of bit-addressable writes. When writing to Port C, the four most significant bits are used as a write protect mask for the least significant bits (0-4, 1-5, 2-6, and 3-7). If the write protect bit is written with a 1, the state of the corresponding output bit is not changed.

Functional Description (Continued)

Ports with Handshake Operation. Ports A and B can be specified as 8-bit input, output, or bidirectional ports with handshake. The CIO provides four different handshakes for its ports: Interlocked, Strobed, Pulsed, and 3-Wire. When specified as a port with handshake, the transfer of data into and out of the port and interrupt generation is under control of the handshake logic. Port C provides the handshake lines as shown in Table 1. Any Port C lines not used for handshake can be used as simple I/O lines or as access lines for Counter/Timer 3.

When Ports A and B are configured as ports with handshake, they are double-buffered. This allows for more relaxed interrupt service routine response time. A second byte can be input to or output from the port before the interrupt for the first byte is serviced. Normally, the Interrupt Pending (IP) bit is set and an interrupt is generated when data is shifted into the Input register (input port) or out of the Output register (output port). For input and output ports, the IP is automatically cleared when the data is read or written. In bidirectional ports, IP is cleared only by command. When the Interrupt on Two Bytes (ITB) control bit is set to 1, interrupts are generated only when two bytes of data are available to be read or written. This allows a minimum of 16 bits of information to be transferred on each interrupt. With ITB set, the IP is not automatically cleared until the second byte of data is read or written.

When the Single Buffer (SB) bit is set to 1, the port acts as if it is only single-buffered. This is useful if the handshake line must be stopped on a byte-by-byte basis.

Ports A and B can be linked to form a 16-bit port by programming a 1 in the Port Link Control (PLC) bit. In this mode, only Port A's Handshake Specification and Command and Status registers are used. Port B must be specified as a bit port. When linked, only Port A has pattern-match capability. Port B's pattern-match capability must be disabled. Also, when the ports are linked, Port B's Data register must be read or written before Port A's.

When a port is specified as a port with handshake, the type of port it is (input, output, or bidirectional) determines the direction of data flow. The data direction for the bidirectional port is determined by a bit in Port C (Table 1). In all cases, the contents of the Data Direction register are ignored. The contents of the Special I/O Control register apply only to output bits (3-state or open-drain). Inputs may not have 1's catchers; therefore, those bits in the Special I/O Control register are ignored. Port C lines used for handshake should be programmed as inputs. The handshake specification overrides Port C's Data Direction register for bits that must be outputs. The contents of Port C's Data Path Polarity register still apply.

Interlocked Handshake. In the Interlocked Handshake mode, the action of the CIO must be acknowledged by the external device before the next action can take place. Figure 7 shows timing for Interlocked Handshake. An output port does not indicate that new data is available until the external device indicates it is ready for the data. Similarly, an input port does not indicate that it is ready for new data until the data source indicates that the previous byte of the data is no longer available, thereby acknowledging the input port's acceptance of the last byte. This allows the CIO to interface directly to the port of a Z8 microcomputer, a UPC, an FIO, an FIFO, or to another CIO port with no external logic.

A 4-bit deskew timer can be inserted in the Data Available ($\overline{DAV}$) output for output ports. As data is transferred to the Buffer register, the deskew timer is triggered. After the number of PCLK cycles specified by the deskew timer time constant plus one, $\overline{DAV}$ is allowed to go Low. The deskew timer therefore guarantees that the output data is valid for a specified minimum amount of time before $\overline{DAV}$

Port A/B Configuration	PC_3	PC_2	PC_1	PC_0
Ports A and B: Bit Ports	Bit I/O	Bit I/O	Bit I/O	Bit I/O
Port A: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O
Port B: Input or Output Port (Interlocked, Strobed, or Pulsed Handshake)*	REQUEST/$\overline{WAIT}$ or Bit I/O	Bit I/O	RFD or $\overline{DAV}$	$\overline{ACKIN}$
Port A or B: Input Port (3-Wire Handshake)	RFD (Output)	$\overline{DAV}$ (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	DAC (Output)
Port A or B: Output Port (3-Wire Handshake)	$\overline{DAV}$ (Output)	DAC (Input)	REQUEST/$\overline{WAIT}$ or Bit I/O	RFD (Input)
Port A or B: Bidirectional Port (Interlocked or Strobed Handshake)	RFD or $\overline{DAV}$	$\overline{ACKIN}$	REQUEST/$\overline{WAIT}$ or Bit I/O	IN/$\overline{OUT}$

*Both Ports A and B can be specified input or output with Interlocked, Strobed, or Pulsed Handshake at the same time if neither uses REQUEST/$\overline{WAIT}$.

Table 1. Port C Bit Utilization

Functional Description (Continued)

goes Low. Deskew timers are available for output ports independent of the type of handshake employed.

Strobed Handshake. In the Strobed Handshake mode, data is "strobed" into or out of the port by the external logic. The falling edge of the Acknowledge Input ($\overline{ACKIN}$) strobes data into or out of the port. Figure 7 shows timing for the Strobed Handshake. In contrast to the Interlocked handshake, the signal indicating the port is ready for another data transfer operates independently of the $\overline{ACKIN}$ input. It is up to the external logic to ensure that data overflows or underflows do not occur.

3-Wire Handshake. The 3-Wire Handshake is designed for the situation in which one output port is communicating with many input ports simultaneously. It is essentially the same as the Interlocked Handshake, except that two signals are used to indicate if an input port is ready for new data or if it has accepted the present data. In the 3-Wire Handshake (Figure 8), the rising edge of one status line indicates that the port is ready for data, and the rising edge of another status line indicates that the data has been accepted. With the 3-Wire Handshake, the output lines of many input ports can be bussed together with open-drain drivers; the output port knows when all the ports have accepted the data and are ready. This is the same handshake as is used on the IEEE-488 bus. Because this handshake requires three lines, only one port (either A or B) can be a 3-Wire Handshake port at a time. The 3-Wire Handshake is not available in the bidirectional mode. Because the port's direction can be changed under software control, however, bidirectional IEEE-488-type transfers can be performed.

Pulsed Handshake. The Pulsed Handshake (Figure 9) is designed to interface to mechanical-type devices that require data to be held for long periods of time and need relatively wide pulses to gate the data into or out of the device. The logic is the same as the Interlocked Handshake mode, except that an internal counter/timer is linked to the handshake logic. If the port is specified in the input mode, the timer is inserted in the $\overline{ACKIN}$ path. The external $\overline{ACKIN}$ input triggers the timer and its output is used as the Interlocked Handshake's normal acknowledge input. If the port is an output port, the timer is placed in the Data Available ($\overline{DAV}$) output path. The timer is triggered when the normal Interlocked Handshake $\overline{DAV}$ output goes Low and the timer output is used as the actual $\overline{DAV}$ output. The counter/timer maintains all of its normal capabilities. This handshake is not available to bidirectional ports.

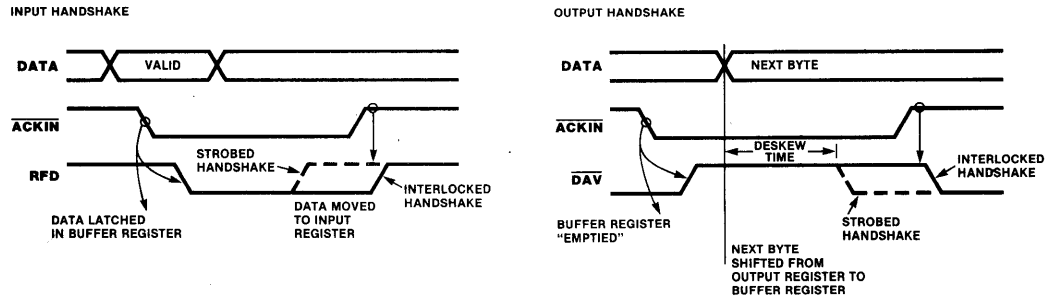


Figure 7. Interlocked and Strobed Handshakes

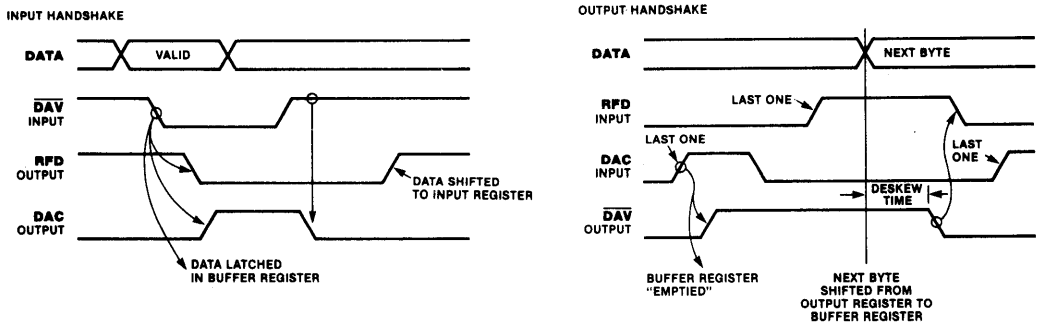


Figure 8. 3-Wire Handshake

Functional Description (Continued)

REQUEST/$\overline{\text{WAIT}}$ Line Operation. Port C can be programmed to provide a status signal output in addition to the normal handshake lines for either Port A or B when used as a port with handshake. The additional signal is either a REQUEST or $\overline{\text{WAIT}}$ signal. The REQUEST signal indicates when a port is ready to perform a data transfer via the CPU interface. It is intended for use with a DMA-type device. The $\overline{\text{WAIT}}$ signal provides synchronization for transfers with a CPU. Three bits in the Port Handshake Specification register provide controls for the REQUEST/$\overline{\text{WAIT}}$ logic. Because the extra Port C line is used, only one port can be specified as a port with a handshake and a REQUEST/$\overline{\text{WAIT}}$ line. The other port must be a bit port.

Operation of the REQUEST line is modified by the state of the port's Interrupt on Two Bytes (ITB) control bit. When ITB is 0, the REQUEST line goes active as soon as the CIO is ready for a data transfer. If ITB is 1, REQUEST does not go active until two bytes can be transferred. REQUEST stays active as long as a byte is available to be read or written.

The SPECIAL REQUEST function is reserved for use with bidirectional ports only. In this case, the REQUEST line indicates the status of the register not being used in the data path at that time. If the IN/$\overline{\text{OUT}}$ line is High, the REQUEST line is High when the Output register is empty. If IN/$\overline{\text{OUT}}$ is Low, the REQUEST line is High when the Input register is full.

Pattern-Recognition Logic Operation. Both Ports A and B can be programmed to generate interrupts when a specific pattern is recognized at the port. The pattern-recognition logic is independent of the port application, thereby allowing the port to recognize patterns in all of its configurations. The pattern can be independently specified for each bit as 1, 0, rising edge, falling edge, or any transition. Individual bits may be masked off. A pattern-match is defined as the simultaneous satisfaction of all nonmasked bit specifications in the AND mode or the satisfaction of any nonmasked bit specifications in either of the OR or OR-Priority Encoded Vector modes.

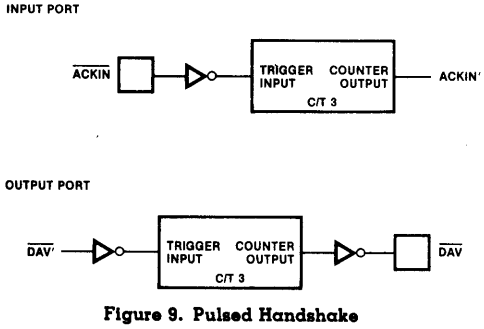


Figure 9. Pulsed Handshake

The pattern specified in the Pattern Definition register assumes that the data path is programmed to be noninverting. If an input bit in the data path is programmed to be inverting, the pattern detected is the opposite of the one specified. Output bits used in the pattern-match logic are internally sampled before the invert/noninvert logic.

Bit Port Pattern-Recognition Operations. During bit port operations, pattern-recognition may be performed on all bits, including those used as I/O for the counter/timers. The input to the pattern-recognition logic follows the value at the pins (through the invert/noninvert logic) in all cases except for simple inputs with 1's catchers. In this case, the output of the 1's catcher is used. When operating in the AND or OR mode, it is the transition from a no-match to a match state that causes the interrupt. In the "OR" mode, if a second match occurs before the first match goes away, it does not cause an interrupt. Since a match condition only lasts a short time when edges are specified, care must be taken to avoid losing a match condition. Bit ports specified in the OR-Priority Encoded Vector mode generate interrupts as long as any match state exists. A transition from a no-match to a match state is not required.

The pattern-recognition logic of bit ports operates in two basic modes: transparent and latched. When the Latch on Pattern Match (LPM) bit is set to 0 (Transparent mode), the interrupt indicates that a specified pattern has occurred, but a read of the Data register does not necessarily indicate the state of the port at the time the interrupt was generated. In the Latched mode (LPM = 1), the state of all the port inputs at the time the interrupt was generated is latched in the input register and held until IP is cleared. In all cases, the PMF indicates the state of the port at the time it is read.

If a match occurs while IP is already set, an error condition exists. If the Interrupt On Error bit (IOE) is 0, the match is ignored. However, if IOE is 1 after the first IP is cleared, it is automatically set to 1 along with the Interrupt Error (ERR) flag. Matches occurring while ERR is set are ignored. ERR is cleared when the corresponding IP is cleared.

When a pattern-match is present in the OR-Priority Encoded Vector mode, IP is set to 1. The IP cannot be cleared until a match is no longer present. If the interrupt vector is allowed to include status, the vector returned during Interrupt Acknowledge indicates the highest priority bit matching its specification at the time of the Acknowledge cycle. Bit 7 is the highest priority and bit 0 is the lowest. The bit initially causing the interrupt may not be the one indicated by the vector if a higher priority bit matches before the Acknowledge. Once the

Functional Description (Continued)

Acknowledge cycle is initiated, the vector is frozen until the corresponding IP is cleared. Where inputs that cause interrupts might change before the interrupt is serviced, the 1's catcher can be used to hold the value. Because a no-match to match transition is not required, the source of the interrupt must be cleared before IP is cleared or else a second interrupt is generated. No error detection is performed in this mode, and the Interrupt On Error bit should be set to 0.

Ports with Handshake Pattern-Recognition Operation. In this mode, the handshake logic normally controls the setting of IP and, therefore, the generation of interrupt requests. The pattern-match logic controls the Pattern-Match Flag (PMF). The data is compared with the match pattern when it is shifted from the Buffer register to the Input register (input port) or when it is shifted from the Output register to the Buffer register (output port). The pattern match logic can override the handshake logic in certain situations. If the port is programmed to interrupt when two bytes of data are available to be read or written, but the first byte matches the specified pattern, the pattern-recognition logic sets IP and generates an interrupt. While PMF is set, IP cannot be cleared by reading or writing the data registers. IP must be cleared by command. The input register is not emptied while IP is set, nor is the output register filled until IP is cleared.

If the Interrupt on Match Only (IMO) bit is set, IP is set only when the data matches the pattern. This is useful in DMA-type application when interrupts are required only after a block of data is transferred.

Counter/Timer Operation. The three independent 16-bit counter/timers consist of a presettable 16-bit down counter, a 16-bit Time Constant register, a 16-bit Current Counter register, an 8-bit Mode Specification register, an 8-bit Command and Status register, and the associated control logic that links these registers.

Function	C/T_1	C/T_2	C/T_3
Counter/Timer Output	PB 4	PB 0	PC 0
Counter Input	PB 5	PB 1	PC 1
Trigger Input	PB 6	PB 2	PC 2
Gate Input	PB 7	PB 3	PC 3

Table 2. Counter/Timer External Access

The flexibility of the counter/timers is enhanced by the provision of up to four lines per counter/timer (counter input, gate input, trigger input, and counter/timer output) for direct external control and status. Counter/Timer 1's external I/O lines are provided by the four most significant bits of Port B. Counter/Timer 2's are provided by the four least significant bits of Port B. Counter/Timer 3's external I/O lines are provided by the four bits of Port C. The utilization of these lines (Table 2) is programmable on a bit-by-bit basis via the Counter/Timer Mode Specification registers.

When external counter/timer I/O lines are to be used, the associated port lines must be vacant and programmed in the proper data direction. Lines used for counter/timer I/O have the same characteristics as simple input lines. They can be specified as inverting or noninverting; they can be read and used with the pattern-recognition logic. They can also include the 1's catcher input.

Counter/Timers 1 and 2 can be linked internally in three different ways. Counter/Timer 1's output (inverted) can be used as Counter/Timer 2's trigger, gate, or counter input. When linked, the counter/timers have the same capabilities as when used separately. The only restriction is that when Counter/Timer 1 drives Counter/Timer 2's count input, Counter/Timer 2 must be programmed with its external count input disabled.

There are three duty cycles available for the timer/counter output: pulse, one-shot, and square-wave. Figure 10 shows the counter/timer waveforms. When the Pulse mode

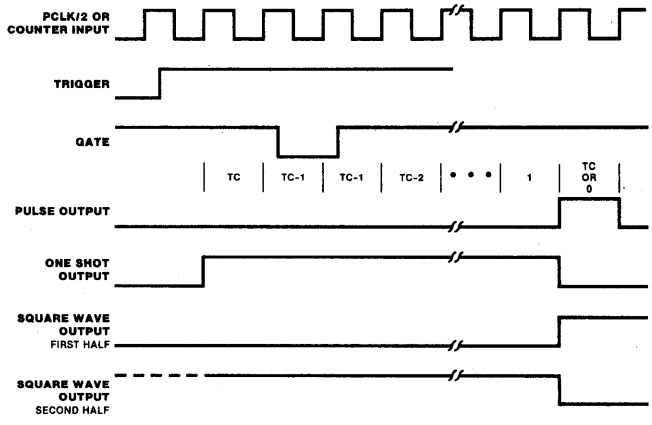


Figure 10. Counter/Timer Waveforms

Functional Description (Continued)

is specified, the output goes High for one clock cycle, beginning when the down-counter leaves the count of 1. In the One-Shot mode, the output goes High when the counter/timer is triggered and goes Low when the down-counter reaches 0. When the square-wave output duty cycle is specified, the counter/timer goes through two full sequences for each cycle. The initial trigger causes the down-counter to be loaded and the normal countdown sequence to begin. If a 1 count is detected on the down-counter's clocking edge, the output goes High and the time constant value is reloaded. On the clocking edge, when both the down-counter and the output are 1's, the output is pulled back Low.

The Continuous/Single Cycle (C/$\overline{SC}$) bit in the Mode Specification register controls operation of the down-counter when it reaches terminal count. If C/$\overline{SC}$ is 0 when a terminal count is reached, the countdown sequence stops. If the C/$\overline{SC}$ bit is 1 each time the countdown counter reaches 1, the next cycle causes the time constant value to be reloaded. The time constant value may be changed by the CPU, and on reload, the new time constant value is loaded.

Counter/timer operations require loading the time constant value in the Time Constant register and initiating the countdown sequence by loading the down-counter with the time constant value. The Time Constant register is accessed as two 8-bit registers. The registers are readable as well as writable, and the access order is irrelevant. A 0 in the Time Constant register specifies a time constant of 65,536. The down-counter is loaded in one of three ways: by writing a 1 to the Trigger Command Bit (TCB) of the Command and Status register, on the rising edge of the external trigger input, or, for Counter/Timer 2 only, on the rising edge of Counter/Timer 1's internal output if the counters are linked via the trigger input. The TCB is write-only, and read always returns 0.

Once the down-counter is loaded, the countdown sequence continues toward terminal count as long as all the counter/timers' hardware and software gate inputs are High. If any of the gate inputs goes Low (0), the countdown halts. It resumes when all gate inputs are 1 again.

The reaction to triggers occurring during a countdown sequence is determined by the state of the Retrigger Enable Bit (REB) in the Mode Specification register. If REB is 0, retriggers are ignored and the countdown continues normally. If REB is 1, each trigger causes the down-counter to be reloaded and the countdown sequence starts over again. If the output is programmed in the Square-Wave mode, retrigger causes the sequence to start over from the initial load of the time constant.

The rate at which the down-counter counts is determined by the mode of the counter/timer. In the Timer mode (the External Count Enable [ECE] bit is 0), the down-counter is clocked internally by a signal that is half the frequency of the PCLK input to the chip. In the Counter mode (ECE is 1), the down-counter is decremented on the rising edge of the counter/timer's counter input.

Each time the counter reaches terminal count, its Interrupt Pending (IP) bit is set to 1, and if interrupts are enabled (IE = 1), an interrupt is generated. If a terminal count occurs while IP is already set, an internal error flag is set. As soon as IP is cleared, it is forced to 1 along with the Interrupt Error (ERR) flag. Errors that occur after the internal flag is set are ignored.

The state of the down-counter can be determined in two ways: by reading the contents of the down-counter via the Current Count register or by testing the Count In Progress (CIP) status bit in the Command and Status register. The CIP status bit is set when the down-counter is loaded; it is reset when the down-counter reaches 0. The Current Count register is a 16-bit register, accessible as two 8-bit registers, which mirrors the contents of the down-counter. This register can be read anytime. However, reading the register is asynchronous to the counter's counting, and the value returned is valid only if the counter is stopped. The down-counter can be reliably read "on the fly" by the first writing of a 1 to the Read Counter Control (RCC) bit in the counter/timer's Command and Status register. This freezes the value in the Current Count register until a read of the least significant byte is performed.

Interrupt Logic Operation. The CIO has five potential sources of interrupts: the three counter/timers and Ports A and B. The priorities of these sources are fixed in the following order: Counter/Timer 3, Port A, Counter/Timer 2, Port B, and Counter/Timer 1. Since the counter/timers all have equal capabilities and Ports A and B have equal capabilities, there is no adverse impact from the relative priorities.

The CIO interrupt priority, relative to other components within the system, is determined by an interrupt daisy chain. Two pins, Interrupt Enable In (IEI) and Interrupt Enable Out (IEO), provide the input and output necessary to implement the daisy chain. When IEI is pulled Low by a higher priority device, the CIO cannot request an interrupt of the CPU. The following discussion assumes that the IEI line is High.

Each source of interrupt in the CIO contains three bits for the control and status of the interrupt logic: an Interrupt Pending (IP) status bit, an Interrupt Under Service (IUS)

Functional Description (Continued)

status bit, and an Interrupt Enable (IE) control bit. IP is set when an event requiring CPU intervention occurs. The setting of IP results in forcing the Interrupt ($\overline{INT}$) output Low, if the associated IE is 1.

The IUS status bit is set as a result of the Interrupt Acknowledge cycle by the CPU and is set only if its IP is of highest priority at the time the Interrupt Acknowledge commences. *It can also be set directly by the CPU. Its* primary function is to control the interrupt daisy chain. When set, it disables lower priority sources in the daisy chain, so that lower priority interrupt sources do not request servicing while higher priority devices are being serviced.

The IE bit provides the CPU with a means of masking off individual sources of interrupts. When IE is set to 1, interrupt is generated normally. When IE is set to 0, the IP bit is set when an event occurs that would normally require service; however, the $\overline{INT}$ output is not forced Low.

The Master Interrupt Enable (MIE) bit allows all sources of interrupts within the CIO to be disabled without having to individually set each IE to 0. If MIE is set to 0, all IPs are masked off and no interrupt can be requested or acknowledged. The Disable Lower Chain (DLC) bit is included to allow the CPU to modify the system daisy chain. When the DLC bit is set to 1, the CIO's IEO is forced Low, independent of the state of the CIO or its IEI input, and all lower priority devices' interrupts are disabled.

As part of the Interrupt Acknowledge cycle, the CIO is capable of responding with an 8-bit interrupt vector that specifies the source of the interrupt. The CIO contains three vector registers: one for Port A, one for Port B, and one shared by the three counter/timers. The vector output is inhibited by setting the No Vector (NV) control bit to 1. The vector output can be modified to include status information to pinpoint more precisely the cause of interrupt. Whether the vector includes status or not is controlled by a Vector Includes Status (VIS) control bit. Each base vector has its own VIS bit and is controlled independently. When MIE = 1, reading the base vector register always includes status, independent of the state of the VIS bit. In this way, all the information obtained by the vector, including status, can be obtained with one additional instruction when VIS is set to 0. When MIE = 0, reading the vector register returns the unmodified base vector so that it can be verified. Another register, the Current Vector register, allows use of the CIO in a polled environment. When read, the data returned is the same as the interrupt vector that would be output in an acknowledge, based on the highest priority IP set. If no unmasked IPs are set, the value FF_H is returned. The Current Vector register is read-only.

Programming

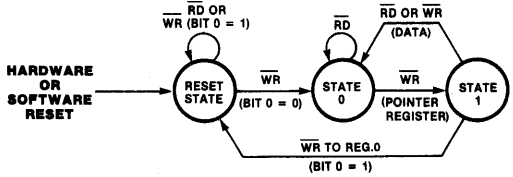
The data registers within the CIO are directly accessed by address lines A_0 and A_1 (Table 3). All other internal registers are accessed by the following two-step sequence, with the address lines specifying a control operation. First, write the address of the target register to an internal 6-bit Pointer Register; then read from or write to the target register. The Data registers can also be accessed by this method.

An internal state machine determines if accesses with A_0 and A_1 equalling 1 are to the Pointer Register or to an internal control register (Figure 11). Following any control read operation, the state machine is in State 0 (the next control access is to the Pointer Register). This can be used to force the state machine into a known state. Control reads in State 0 return the contents of the last register pointed to. Therefore, a register can be read continuously without writing to the Pointer. While the CIO is in State 1 (next control access is to the register pointed to), many internal operations are suspended—no IPs are set and internal status is frozen. Therefore, to minimize interrupt latency and to allow continuous status updates, the CIO should not be left in State 1.

The CIO is reset by forcing $\overline{RD}$ and $\overline{WR}$ Low simultaneously (normally an illegal condition) or by writing a 1 to the Reset bit. Reset disables all functions except a read from or write to the Reset bit; writes to all other bits are ignored, and all reads return 01_H. In this state, all control bits are forced to 0 and may be programmed only after clearing the Reset bit (by writing a 0 to it).

A_1	A_0	Register
0	0	Port C's Data Register
0	1	Port B's Data Register
1	0	Port A's Data Register
1	1	Control Registers

Table 3. Register Selection



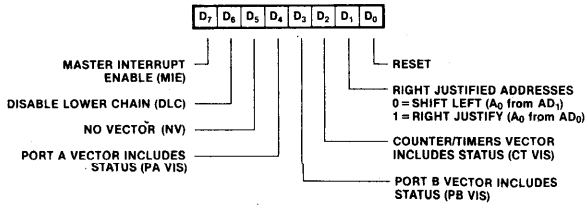
NOTE: State changes occur only when $A_0 = A_1 = 1$. No other accesses have effect.

Figure 11. State Machine Operation

Registers

Master Interrupt Control Register

Address: 000000
(Read/Write)



Master Configuration Control Register

Address: 000001
(Read/Write)

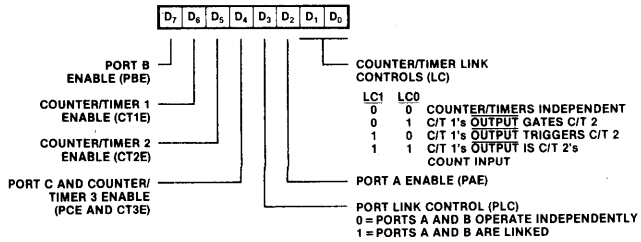
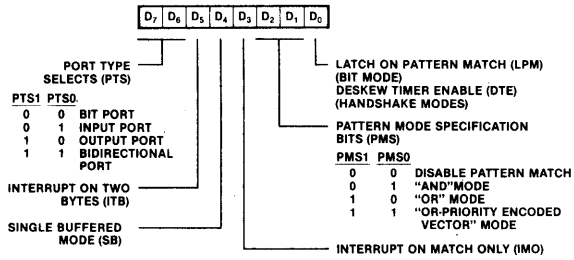


Figure 12. Master Control Registers

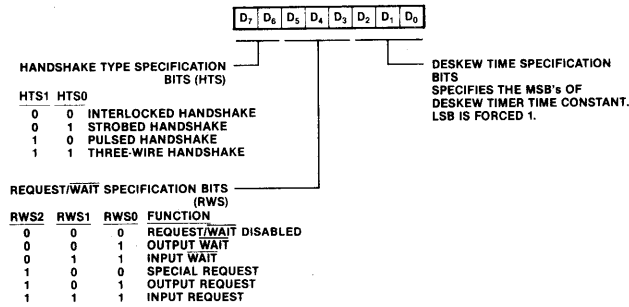
Port Mode Specification Registers

Addresses: 100000 Port A
101000 Port B
(Read/Write)



Port Handshake Specification Registers

Addresses: 100001 Port A
101001 Port B
(Read/Write)



Port Command and Status Registers

Addresses: 001000 Port A
001001 Port B
(Read/Partial Write)

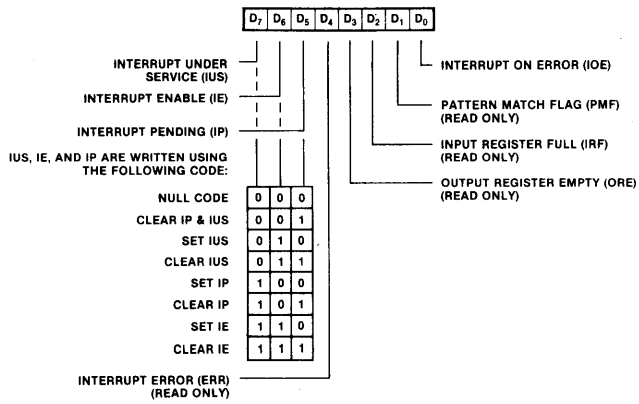
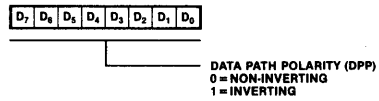


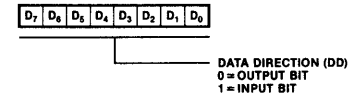
Figure 13. Port Specifications Registers

Registers
(Continued)

Data Path Polarity Registers
Addresses: 100010 Port A
101010 Port B
000101 Port C (4 LSBs only)
(Read/Write)



Data Direction Registers
Addresses: 100011 Port A
101011 Port B
000110 Port C (4 LSBs only)
(Read/Write)



Special I/O Control Registers
Addresses: 100100 Port A
101100 Port B
000111 Port C (4 LSBs only)
(Read/Write)

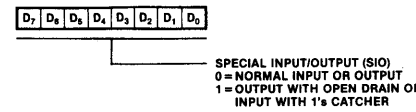
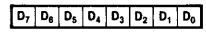


Figure 14. Bit Path Definition Registers

Port Data Registers
Addresses: 001101 Port A*
001110 Port B*
(Read/Write)



*These registers can be addressed directly.

Port C Data Register
Address: 001111*
(Read/Write)

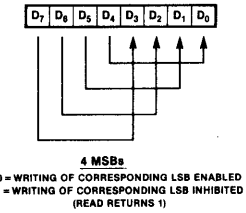
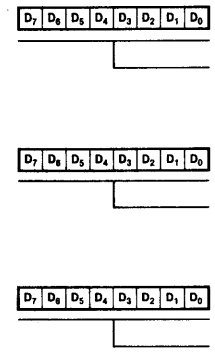


Figure 15. Port Data Registers

Pattern Polarity Registers (PP)
Addresses: 100101 Port A
101101 Port B
(Read/Write)

Pattern Transition Registers (PT)
Addresses: 100110 Port A
101110 Port B
(Read/Write)

Pattern Mask Registers (PM)
Addresses: 100111 Port A
101111 Port B
(Read/Write)



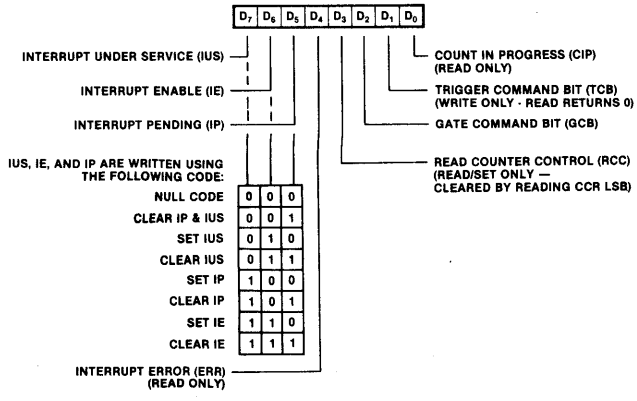
PM	PT	PP	PATTERN SPECIFICATION
0	0	X	BIT MASKED OFF
0	1	X	ANY TRANSITION
1	0	0	ZERO
1	0	1	ONE
1	1	0	ONE TO ZERO TRANSITION (↓)
1	1	1	ZERO-TO-ONE TRANSITION (↑)

Figure 16. Pattern Definition Registers

Registers
(Continued)

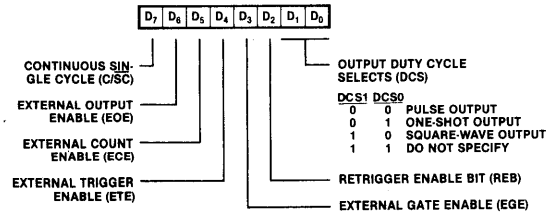
Counter/Timer Command and Status Registers

Addresses: 001010 Counter/Timer 1
001011 Counter/Timer 2
001100 Counter/Timer 3
(Read/Partial Write)



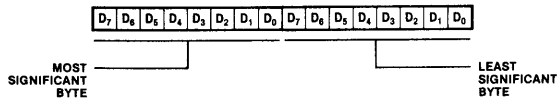
Counter/Timer Mode Specification Registers

Addresses: 011100 Counter/Timer 1
011101 Counter/Timer 2
011110 Counter/Timer 3
(Read/Write)



Counter/Timer Current Count Registers

Addresses: 010000 Counter/Timer 1's MSB
010001 Counter/Timer 1's LSB
010010 Counter/Timer 2's MSB
010011 Counter/Timer 2's LSB
010100 Counter/Timer 3's MSB
010101 Counter/Timer 3's LSB
(Read Only)



Counter/Timer Time Constant Registers

Addresses: 010110 Counter/Timer 1's MSB
010111 Counter/Timer 1's LSB
011000 Counter/Timer 2's MSB
011001 Counter/Timer 2's LSB
011010 Counter/Timer 3's MSB
011011 Counter/Timer 3's LSB
(Read/Write)

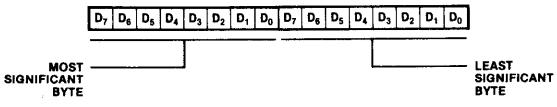
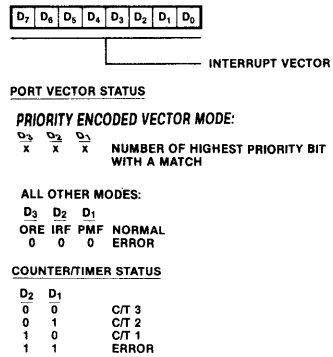


Figure 17. Counter/Timer Registers

Registers (Continued)

Interrupt Vector Register
Addresses: 000010 Port A
000011 Port B
000100 Counter/Timers
(Read/Write)



Current Vector Register
Address: 011111
(Read only)

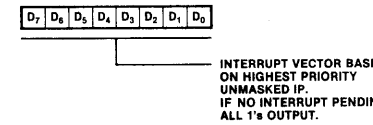


Figure 18. Interrupt Vector Registers

Register Address Summary

Main Control Registers

Address	Register Name
000000	Master Interrupt Control
000001	Master Configuration Control
000010	Port A's Interrupt Vector
000011	Port B's Interrupt Vector
000100	Counter/Timer's Interrupt Vector
000101	Port C's Data Path Polarity
000110	Port C's Data Direction
000111	Port C's Special I/O Control

Most Often Accessed Registers

Address	Register Name
001000	Port A's Command and Status
001001	Port B's Command and Status
001010	Counter/Timer 1's Command and Status
001011	Counter/Timer 2's Command and Status
001100	Counter/Timer 3's Command and Status
001101	Port A's Data (can be accessed directly)
001110	Port B's Data (can be accessed directly)
001111	Port C's Data (can be accessed directly)

Counter/Timer Related Registers

Address	Register Name
010000	Counter/Timer 1's Current Count-MSBs
010001	Counter/Timer 1's Current Count-LSBs
010010	Counter/Timer 2's Current Count-MSBs
010011	Counter/Timer 2's Current Count-LSBs
010100	Counter/Timer 3's Current Count-MSBs
010101	Counter/Timer 3's Current Count-LSBs
010110	Counter/Timer 1's Time Constant-MSBs
010111	Counter/Timer 1's Time Constant-LSBs
011000	Counter/Timer 2's Time Constant-MSBs
011001	Counter/Timer 2's Time Constant-LSBs
011010	Counter/Timer 3's Time Constant-MSBs
011011	Counter/Timer 3's Time Constant-LSBs
011100	Counter/Timer 1's Mode Specification
011101	Counter/Timer 2's Mode Specification
011110	Counter/Timer 3's Mode Specification
011111	Current Vector

Port A Specification Registers

Address	Register Name
100000	Port A's Mode Specification
100001	Port A's Handshake Specification
100010	Port A's Data Path Polarity
100011	Port A's Data Direction
100100	Port A's Special I/O Control
100101	Port A's Pattern Polarity
100110	Port A's Pattern Transition
100111	Port A's Pattern Mask

Port B Specification Registers

Address	Register Name
101000	Port B's Mode Specification
101001	Port B's Handshake Specification
101010	Port B's Data Path Polarity
101011	Port B's Data Direction
101100	Port B's Special I/O Control
101101	Port B's Pattern Polarity
101110	Port B's Pattern Transition
101111	Port B's Pattern Mask

Timing

Read Cycle. At the beginning of a read cycle, the CPU places an address on the address bus. Bits A_0 and A_1 specify a CIO register; the remaining address bits and status information are combined and decoded to generate a Chip Enable ($\overline{CE}$) signal that selects the CIO. When Read ($\overline{RD}$) goes Low, data from the specified register is gated onto the data bus.

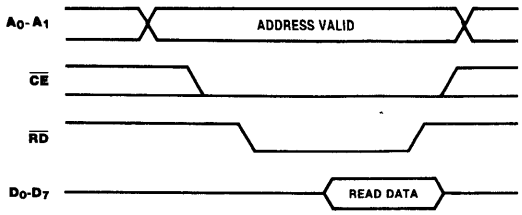


Figure 19. Read Cycle Timing

Write Cycle. At the beginning of a write cycle, the CPU places an address on the data bus. Bits A_0 and A_1 specify a CIO register; the remaining address bits and status information are combined and decoded to generate a Chip Enable ($\overline{CE}$) signal that selects the CIO. When $\overline{WR}$ goes Low, data placed on the bus by the CPU is strobed into the specified CIO register.

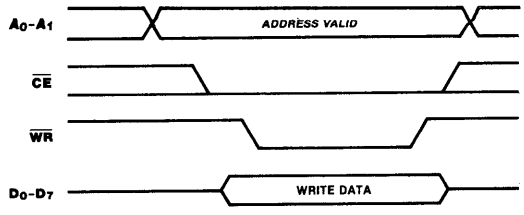


Figure 20. Write Cycle Timing

Interrupt Acknowledge. The CIO pulls its Interrupt Request ($\overline{INT}$) line Low, requesting interrupt service from the CPU, if an Interrupt Pending (IP) bit is set and interrupts are enabled. The CPU responds with an Interrupt Acknowledge cycle. When Interrupt Acknowledge ($\overline{INTACK}$) goes true and the IP is set, the CIO forces Interrupt Enable Out (IEO) Low, disabling all lower priority devices in the interrupt daisy chain. If the CIO is the highest priority device requesting service (IEI is High), it places its interrupt vector on the data bus and sets the Interrupt Under Service (IUS) bit when Read ($\overline{RD}$) goes Low.

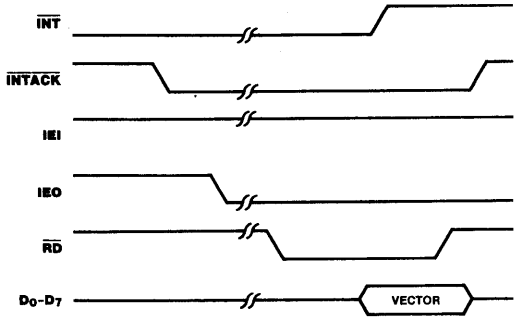


Figure 21. Interrupt Acknowledge Timing

Absolute Maximum Ratings

Voltages on all inputs and outputs with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . As Specified in Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the referenced pin. Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- T_A as specified in Ordering Information

All ac parameters assume a load capacitance of 50 pF max.

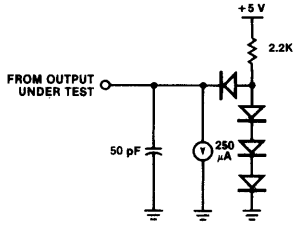


Figure 22. Standard Test Load

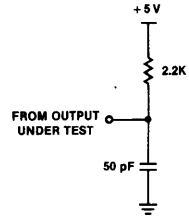


Figure 23. Open-Drain Test Load

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{IH}	Input High Voltage	2.0	V_{CC} + 0.3	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
			0.5	V	I_{OL} = +3.2 mA
I_{IL}	Input Leakage		±10.0	µA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{OL}	Output Leakage		±10.0	µA	0.4 ≤ V_{OUT} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		200	mA	

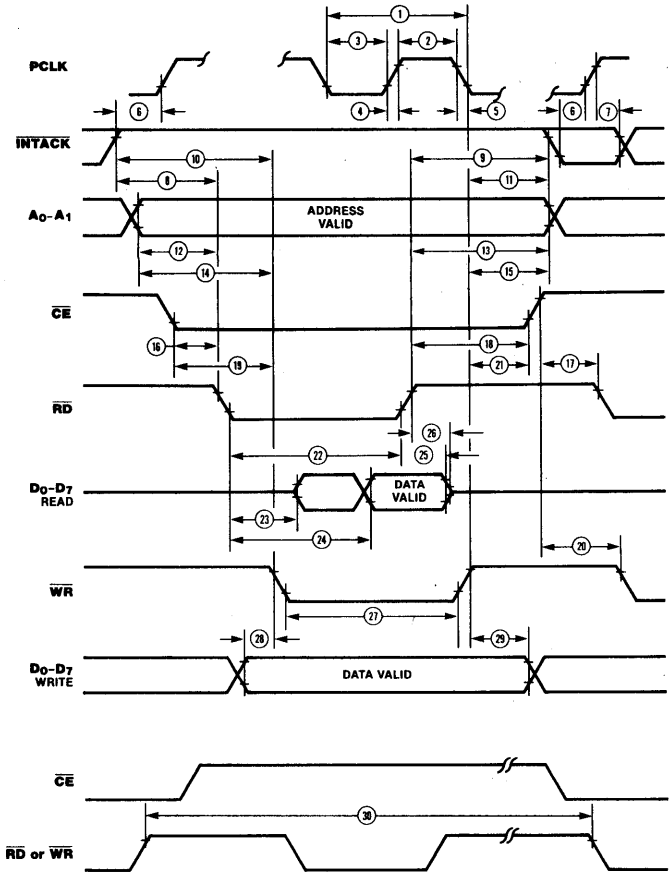
V_{CC} = 5 V ± 5% unless otherwise specified, over specified temperature range.

Capacitance

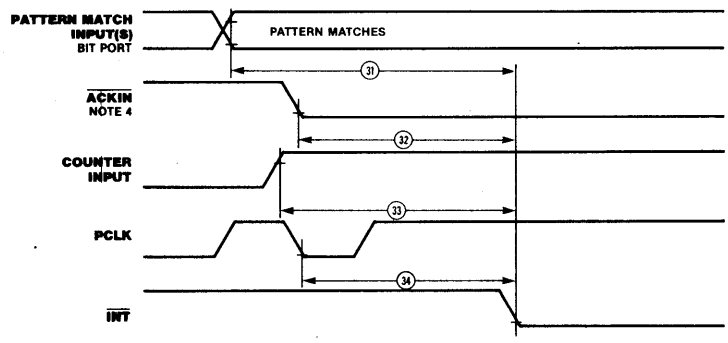
Symbol	Parameter	Min	Max	Unit	Test Condition
C_{IN}	Input Capacitance		10	pF	Unmeasured Pins Returned to Ground
C_{OUT}	Output Capacitance		15	pF	
$C_{I/O}$	Bidirectional Capacitance		20	pF	

f = 1 MHz, over specified temperature range.

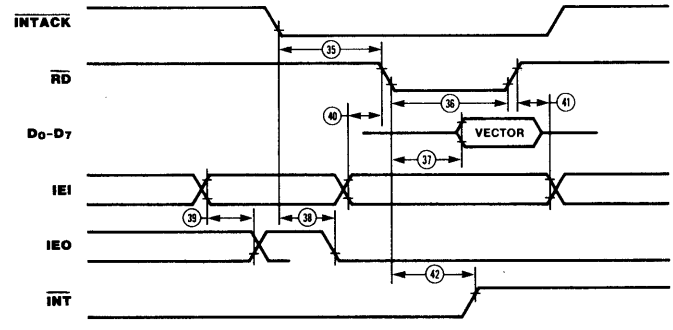
CPU Interface Timing



Interrupt Timing



Interrupt Acknowledge Timing



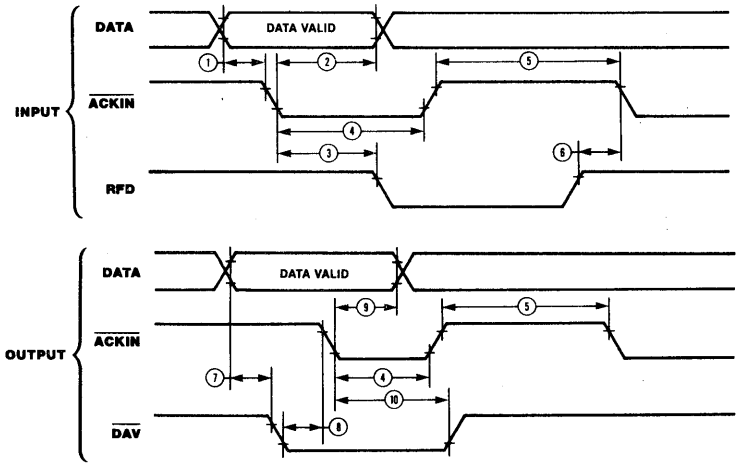
No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TcPC	PCLK Cycle time	250	4000	165	4000	
2	TwPCh	PCLK Width (High)	105	2000	70	2000	
3	TwPCl	PCLK Width (Low)	105	2000	70	2000	
4	TrPC	PCLK Rise Time		20		10	
5	TfPC	PCLK Fall Time		20		15	
6	TsIA(PC)	$\overline{INTACK}$ to PCLK ↑ Setup Time	100		100		
7	ThIA(PC)	$\overline{INTACK}$ to PCLK ↑ Hold Time	0		0		
8	TsIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↓ Setup Time	200		200		
9	ThIA(RD)	$\overline{INTACK}$ to $\overline{RD}$ ↑ Hold Time	0		0		
10	TsIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↓ Setup Time	200		200		
11	ThIA(WR)	$\overline{INTACK}$ to $\overline{WR}$ ↑ Hold Time	0		0		
12	TsA(RD)	Address to $\overline{RD}$ ↓ Setup Time	80		80		
13	ThA(RD)	Address to $\overline{RD}$ ↑ Hold Time	0		0		
14	TsA(WR)	Address to $\overline{WR}$ ↓ Setup Time	80		80		
15	ThA(WR)	Address to $\overline{WR}$ ↑ Hold Time	0		0		
16	TsCEl(RD)	$\overline{CE}$ Low to $\overline{RD}$ ↓ Setup Time	0		0		1
17	TsCEh(RD)	$\overline{CE}$ High to $\overline{RD}$ ↓ Setup Time	100		70		1
18	ThCE(RD)	$\overline{CE}$ to $\overline{RD}$ ↑ Hold Time	0		0		1
19	TsCEl(WR)	$\overline{CE}$ Low to $\overline{WR}$ ↓ Setup Time	0		0		
20	TsCEh(WR)	$\overline{CE}$ High to $\overline{WR}$ ↓ Setup Time	100		70		
21	ThCE(WR)	$\overline{CE}$ to $\overline{WR}$ ↑ Hold Time	0		0		
22	TwRDl	$\overline{RD}$ Low Width	390		250		1
23	TdRD(DRA)	$\overline{RD}$ ↓ to Read Data Active Delay	0		0		
24	TdRDf(DR)	$\overline{RD}$ ↓ to Read Data Valid Delay		255		180	
25	TdRDr(DR)	$\overline{RD}$ ↑ to Read Data Not Valid Delay	0		0		
26	TdRD(DRz)	$\overline{RD}$ ↑ to Read Data Float Delay		70		45	2
27	TwWRl	$\overline{WR}$ Low Width	390		250		
28	TsDW(WR)	Write Data to $\overline{WR}$ ↓ Setup Time	0		0		
29	ThDW(WR)	Write Data to $\overline{WR}$ ↑ Hold Time	0		0		
30	Trc	Valid Access Recovery Time	1000*		650		3
31	TdPM(INT)	Pattern Match to $\overline{INT}$ Delay (Bit Port)		2+800		2	6
32	TdACK(INT)	ACKIN to $\overline{INT}$ Delay (Port with Handshake)		10+600		10	4,6
33	TdCI(INT)	Counter Input to $\overline{INT}$ Delay (Counter Mode)		2+700		2	6
34	TdPC(INT)	PCLK to $\overline{INT}$ Delay (Timer Mode)		3+700		3	6
35	TsIA(RDA)	$\overline{INTACK}$ to $\overline{RD}$ ↓ (Acknowledge) Setup Time	350		250		5
36	TwRDA	$\overline{RD}$ (Acknowledge) Width	350		250		
37	TdRDA(DR)	$\overline{RD}$ ↓ (Acknowledge) to Read Data Valid Delay		250		180	
38	TdIA(IEO)	$\overline{INTACK}$ ↓ to IEO ↓ Delay		350		250	5
39	TdIEI(IEO)	IEI to IEO Delay		150		100	5
40	TsIEI(RDA)	IEI to $\overline{RD}$ ↓ (Acknowledge) Setup Time	100		70		5
41	ThIEI(RDA)	IEI to $\overline{RD}$ ↑ (Acknowledge) Hold Time	100		70		
42	TdRDA(INT)	$\overline{RD}$ ↓ (Acknowledge) to $\overline{INT}$ ↑ Delay		600		600	

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. Float delay is measured to the time when the output has changed 0.5 V with minimum ac load and maximum dc load.
3. Trc is the specified number or 3 TcPC, whichever is longer.
4. The delay is from DAV ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.
5. The parameters for the devices in any particular daisy chain must meet the following constraint: The delay from INTACK ↓ to RD ↓ must be greater than the sum of TdIA(IEO) for the highest priority peripheral, TsIEI(RDA) for the lowest priority peripheral, and TdIEI(IEO) for each peripheral separating them in the chain.
6. Units are equal to TcPC plus ns.

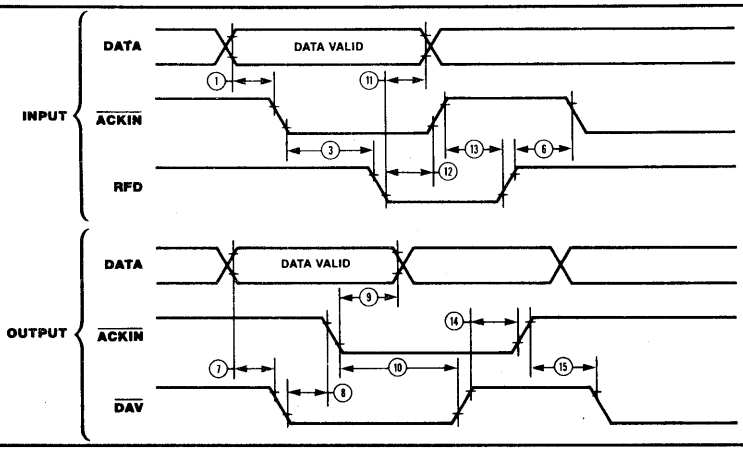
* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Units in nanoseconds (ns), except as noted.

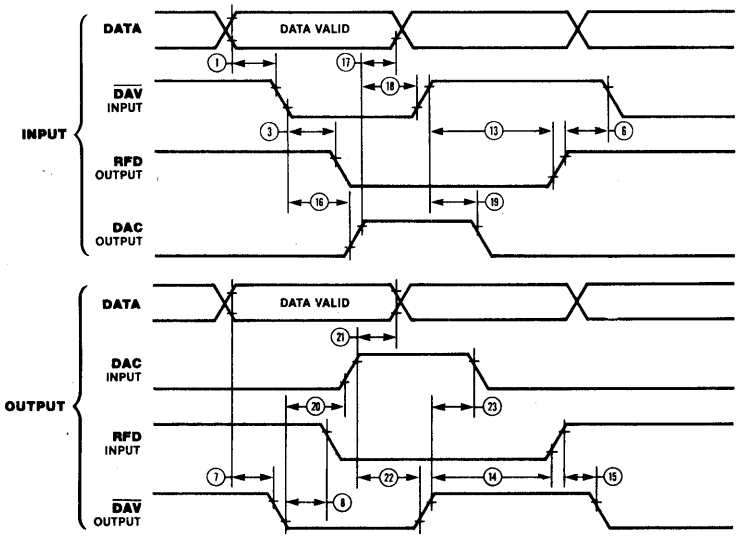
Strobed Handshake



Interlocked Handshake



3-Wire Handshake

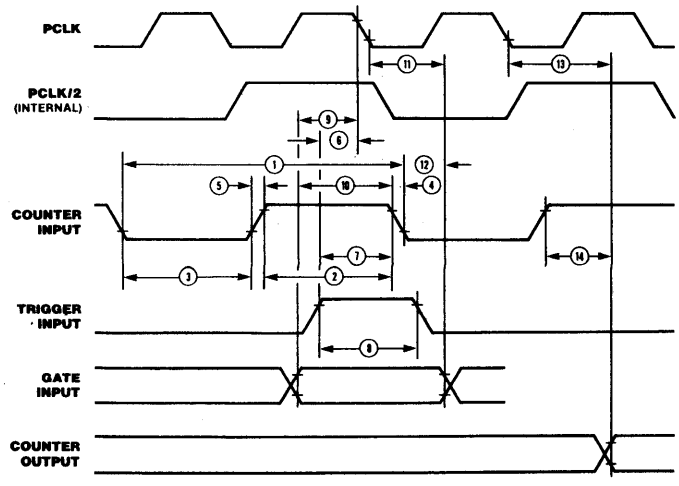


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TsDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Setup Time	0		0		
2	ThDI(ACK)	Data Input to $\overline{ACKIN}$ ↓ Hold Time—Strobed Handshake					
3	TdACKf(RFD)	$\overline{ACKIN}$ ↓ to RFD ↓ Delay	0		0		
4	TwACKl	$\overline{ACKIN}$ Low Width—Strobed Handshake					
5	TwACKh	$\overline{ACKIN}$ High Width—Strobed Handshake					
6	TdRFDr(ACK)	RFD ↑ to $\overline{ACKIN}$ ↓ Delay	0		0		
7	TsDO(DAV)	Data Out to $\overline{DAV}$ ↓ Setup Time	25		20		1
8	TdDAVf(ACK)	$\overline{DAV}$ ↓ to $\overline{ACKIN}$ ↓ Delay	0		0		
9	ThDO(ACK)	Data Out to $\overline{ACKIN}$ ↓ Hold Time	2		2		2
10	TdACK(DAV)	$\overline{ACKIN}$ ↓ to $\overline{DAV}$ ↑ Delay	2		2		2
11	THDI(RFD)	Data Input to RFD ↓ Hold Time—Interlocked Handshake					
12	TdRFDf(ACK)	RFD ↓ to $\overline{ACKIN}$ ↑ Delay Interlocked Handshake	0		0		
13	TdACKr(RFD)	$\overline{ACKIN}$ ↑ ($\overline{DAV}$ ↑) to RFD ↑ Delay—Interlocked and 3-Wire Handwshake	0		0		
14	TdDAVr(ACK)	$\overline{DAV}$ ↑ to $\overline{ACKIN}$ ↑ (RFD ↑)—Interlocked and 3-Wire Handshake	0		0		
15	TdACK(DAV)	$\overline{ACKIN}$ ↑ (RFD ↑) to $\overline{DAV}$ ↓ Delay—Interlocked and 3-Wire Handshake	0		0		
16	TdDAVlf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay—Input 3-Wire Handshake	0		0		
17	ThDI(DAC)	Data Input to DAC ↑ Hold Time—3-Wire Handshake	0		0		
18	TdDACOr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay—Input 3-Wire Handshake	0		0		
19	TdDAVIr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay—Input 3-Wire Handshake	0		0		
20	TdDAVOf(DAC)	$\overline{DAV}$ ↓ to DAC ↑ Delay—Output 3-Wire Handshake	0		0		
21	ThDO(DAC)	Data Output to DAC ↑ Hold Time—3-Wire Handshake	2		2		2
22	TdDACIr(DAV)	DAC ↑ to $\overline{DAV}$ ↑ Delay—Output 3-Wire Handshake	2		2		2
23	TdDAVOr(DAC)	$\overline{DAV}$ ↑ to DAC ↓ Delay—Output 3-Wire Handshake	0		0		

NOTES:
1. This time can be extended through the use of deskew timers.
2. Units equal to TcPC.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Counter/ Timer Timing

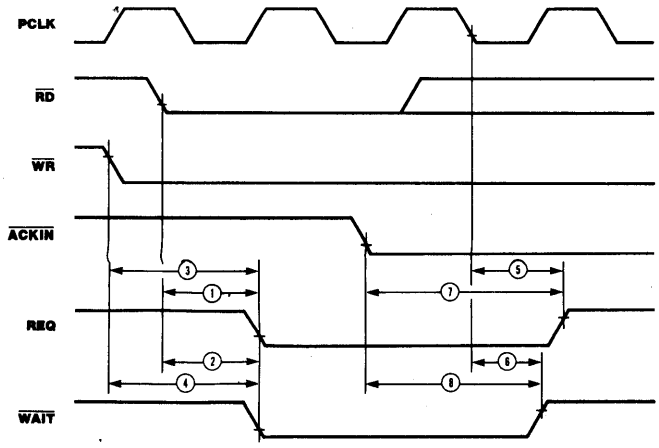


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TcCI	Counter Input Cycle Time	500		330		
2	TCIh	Counter Input High Width	230		150		
3	TWCIl	Counter Input Low Width	230		150		
4	TfCI	Counter Input Fall Time		20		15	
5	TrCI	Counter Input Rise Time		20		15	
6	TsTI(PC)	Trigger Input to PCLK ↓ Setup Time (Timer Mode)					1
7	TsTI(CI)	Trigger Input to Counter Input ↓ Setup Time (Counter Mode)					1
8	TwTI	Trigger Input Pulse Width (High or Low)					
9	TsGI(PC)	Gate Input to PCLK ↓ Setup Time (Timer Mode)					1
10	TsGI(CI)	Gate Input to Counter Input ↓ Setup Time (Counter Mode)					1
11	ThGI(PC)	Gate Input to PCLK ↓ Hold Time (Timer Mode)					1
12	ThGI(CI)	Gate Input to Counter Input ↓ Hold Time (Counter Mode)					1
13	TdPC(CO)	PCLK to Counter Output Delay (Timer Mode)					
14	TdCI(CO)	Counter Input to Counter Output Delay (Counter Mode)					

NOTES:
1. These parameters must be met to guarantee trigger or gate are valid for the next counter/timer cycle.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns).

REQUEST/ WAIT Timing

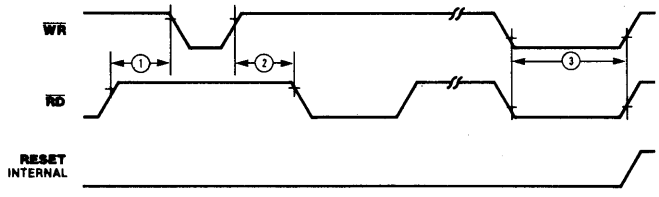


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRD(REQ)	$\overline{RD}$ ↓ to REQ ↓ Delay		500			
2	TdRD(WAIT)	$\overline{RD}$ ↓ to $\overline{WAIT}$ ↓ Delay		500			
3	TdWR(REQ)	$\overline{WR}$ ↓ to REQ ↓ Delay		500			
4	TdWR(WAIT)	$\overline{WR}$ ↓ to $\overline{WAIT}$ ↓ Delay		500			
5	TdPC(REQ)	PCLK ↓ to REQ ↑ Delay		300			
6	TdPC(WAIT)	PCLK ↓ to $\overline{WAIT}$ ↑ Delay		300			
7	TdACK(REQ)	$\overline{ACKIN}$ ↓ to REQ ↑ Delay		8+100			1,2
8	TdACK(WAIT)	$\overline{ACKIN}$ ↓ to $\overline{WAIT}$ ↑ Delay		10+600			1,2

NOTES:
1. The delay is fronm DAV ↓ for 3-Wire Input Handshake. The delay is from DAC ↑ for 3-Wire Output Handshake.
2. Units equal to TcPC + ns.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".
† Units in nanoseconds (ns), except as noted.

Reset Timing

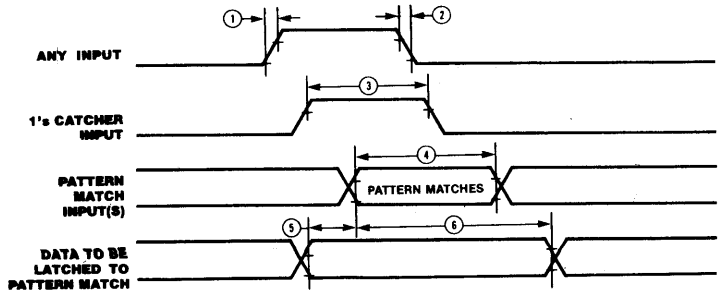


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TdRD(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓ for No Reset	50		50		
2	TdWR(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓ for No Reset	50		50		
3	TwRES	Minimum Width of $\overline{RD}$ and $\overline{WR}$ both Low for Reset	250		250		

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0".

† Unites in nanoseconds (ns).

Miscellaneous Port Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
1	TrI	Any Input Rise Time		100		100	
2	TfI	Any Input Fall Time		100		100	
3	Tw1's	1's Catcher High Width	250		170		1
4	TwPM	Pattern Match Input Valid (Bit Port)	750		500		
5	TsPMD	Data Latched on Pattern Match Setup Time (Bit Port)	0		0		
6	ThPMD	Data Latched on Pattern Match Hold Time (Bit Port)	1000		650		

NOTES:
1. If the input is programmed inverting, a Low-going pulse of the same width will be detected.

* Timings are preliminary and subject to change. All timing references assume 2.0 V for a logic "1" and 0.8 V for a logig "0"
† Units in nanoseconds (ns).

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8536	CE	4.0 MHz	CIO (40-pin)	Z8536A	CE	6.0 MHz	CIO (40-pin)
Z8536	CM	4.0 MHz	Same as above	Z8536A	CM	6.0 MHz	Same as above
Z8536	CMB	4.0 MHz	Same as above	Z8536A	CMB	6.0 MHz	Same as above
Z8536	CS	4.0 MHz	Same as above	Z8536A	CS	6.0 MHz	Same as above
Z8536	DE	4.0 MHz	Same as above	Z8536A	DE	6.0 MHz	Same as above
Z8536	DS	4.0 MHz	Same as above	Z8536A	DS	6.0 MHz	Same as above
Z8536	PE	4.0 MHz	Same as above	Z8536A	PE	6.0 MHz	Same as above
Z8536	PS	4.0 MHz	Same as above	Z8536A	PS	6.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, M = -55°C to 125°C, MB = -55°C to 125°C with MIL-STD-883 with Class B processing, S = 0°C to +70°C.

Z8581 Clock Generator and Controller

Zilog

Product Specification

September 1983

FEATURES

- Two independent 20 MHz oscillators generate two 10 MHz clock outputs and one 20 MHz clock output.
- Oscillator input frequency sources can be either crystals or external oscillators.
- Outputs directly drive the Z80 and Z8000 microprocessor clock inputs.
- Can be used as a general-purpose clock generator.
- 18-pin slimline package used; single +5 V dc power required.
- Provides ability to stretch High and/or Low phase of clock signal under external control.
  - On-chip 2-bit counter can be used to selectively stretch clock cycles.
- On-chip reset logic
  - Reset output is synchronized with System Clock output.
  - Power-up reset period is maintained for a minimum of 30 ms.
  - External input initiates system reset.

GENERAL DESCRIPTION

The Z8581 Clock Generator and Controller is a versatile addition to Zilog's family of Universal microprocessor components. The selective clock-stretching capabilities and variety of timing outputs produced by this device allow it to easily meet the timing design requirements of systems with microprocessors and LSI peripherals. The clock output drivers of the Z8581 also meet the non-TTL voltage requirements for driving NMOS clock inputs with no additional external components. The Z8581 provides an elegant, single-chip solution to the design of system clocks for microprocessor-based products.

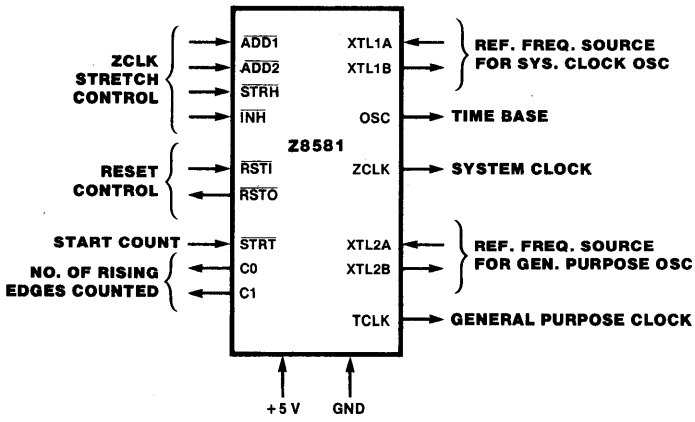


Figure 1. Pin Functions

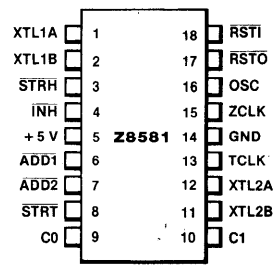


Figure 2. Pin Assignments

The Z8581 is packaged in an 18-pin slimline DIP (see Figure 1 for pin functions and Figure 2 for pin assignments).

The Z8581 oscillators are referenced as the system clock oscillator and the general-purpose clock oscillator. Both oscillators are driven by external crystals or other frequency sources.

System Clock Oscillator. The timing outputs provided by this oscillator consist of a Time Base output (OSC), at the frequency of the reference source, and a stretchable System Clock output (ZCLK), at a frequency determined by the stretch control inputs. An on-chip TTL driver at OSC and an NMOS driver at ZCLK eliminate the need for external buffers or drivers. The NMOS drivers can drive 200 pF loads to within 0.2 volts of V_{CC} and sink 8 mA. Output rise and fall times are 10 ns maximum.

ZCLK can be stretched under program or hardwired control by selectively adding periods equivalent to a full OSC cycle to either the High or Low portion of a clock cycle. One, two, or three periods can be added to double, triple, or quadruple the duration of the selected ZCLK half-cycle. Adding periods to ZCLK is a function of the ADD1 and ADD2 inputs. These active Low inputs are sampled prior to the rising edge of signal OSC; their sampled status represents the number of periods to be added to ZCLK.

Two additional control inputs, $\overline{INH}$ and $\overline{STRH}$, affect the stretch function. Input $\overline{INH}$, when asserted, inhibits the function of $\overline{ADD1}$ and $\overline{ADD2}$. Input $\overline{STRH}$ stretches the ZCLK output for as long as it is asserted (Low); it overrides all other stretch control inputs.

Table 1 summarizes the functions performed by the stretch control inputs.

The system clock oscillator also contains a 2-bit ZCLK counter. This counter, when initialized by the assertion of $\overline{STRT}$, counts the next four rising edges of the ZCLK output. The current count is presented on outputs C0 and C1. This counter and its outputs enable the user to determine the occurrence (rising edge) of each of four clocks after a specific event ($\overline{STRT}$ is asserted). This facility can, for example, be used to determine when a delay is to be inserted into a CPU machine cycle when $\overline{STRT}$ is triggered by either an $\overline{M1}$ (Z80) or an $\overline{AS}$ (Z8000) input signal.

Table 1. Stretch Control Functions

$\overline{STRH}$	$\overline{INH}$	$\overline{ADD2}$	$\overline{ADD1}$	Periods Added
0	X	X	X	Unlimited
1	0	X	X	0
1	1	0	0	3
1	1	0	1	2
1	1	1	0	1
1	1	1	1	0

Notes: X = Don't Care, 1 = High, 0 = Low

The clock stretch capability allows systems to run at the nominal high speed of ZCLK, except during cycles that require more time than usual to complete a transaction. For example, extended access time may be required in accessing certain areas of memory, in accessing I/O devices, or in other CPU/Peripheral transactions. Figures 3 and 4 illustrate, respectively, the circuit configuration and timing required to stretch the Z8000 Address Strobe ($\overline{AS}$) and Data Strobe ($\overline{DS}$) to allow more time for address functions and to enable the CPU to operate with memories that have a relatively long access time.

In addition, the ZCLK stretch control logic can be hardwired to meet various duty cycle requirements. For example, a simple hardwired connection can cause every other ZCLK cycle to be stretched to produce a ZCLK output with a 33% duty cycle.

The system clock oscillator also provides a system reset output ($\overline{RSTO}$) that is synchronized with ZCLK. This output is controlled by a system reset input ($\overline{RSTI}$) during normal system reset operations and by delay circuitry in the system clock oscillator during power-up operations. During a normal system reset operation, a Low on $\overline{RSTI}$ causes $\overline{RSTO}$ to be asserted (Low) on the next rising edge of ZCLK. Output $\overline{RSTO}$ is held Low for a period of 16 ZCLK clock cycles (the required reset time for both the Z80 and Z8000 CPU system reset functions). During a power-up operation, $\overline{RSTO}$ is asserted for a minimum of 30 ms after power is turned on (the time required for both the Z80 and Z8000 power-up functions).

General-Purpose Oscillator. This oscillator provides a fixed frequency General-Purpose Clock output (TCLK) at half its source frequency. This output is useful for system timing functions such as controlling a baud rate generator. Output TCLK can also be used as the frequency reference source for the system clock oscillator.

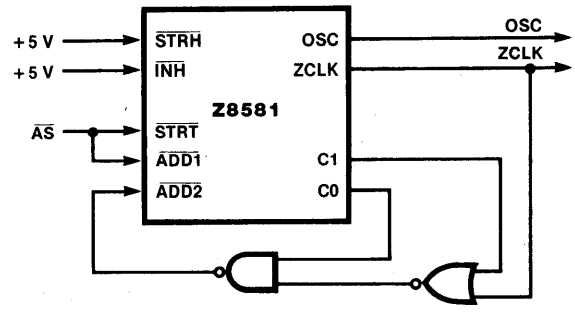


Figure 3. Configuration for Stretching Z8000 Address ($\overline{AS}$) and Data ($\overline{DS}$) Strobes

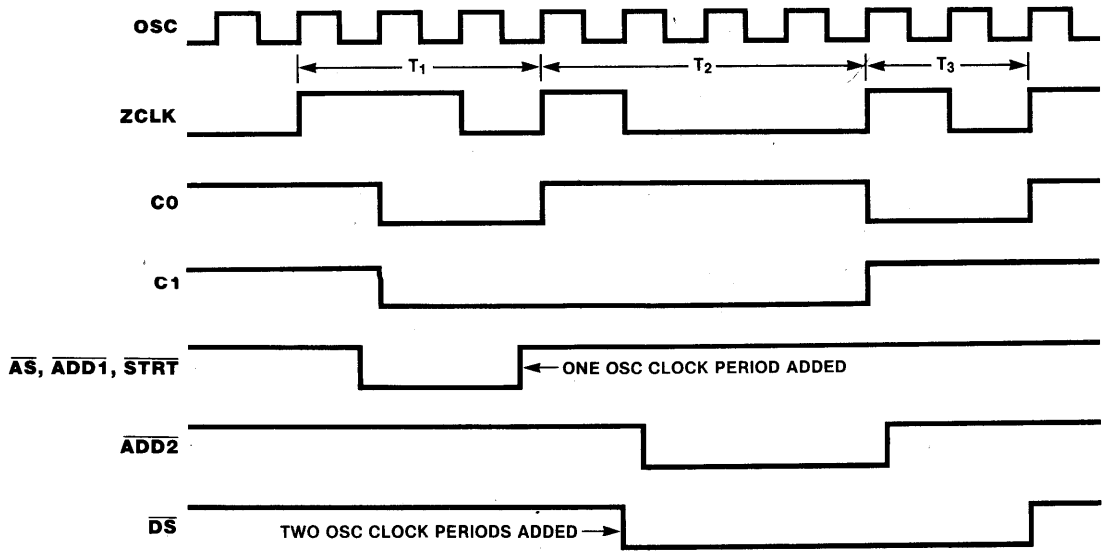


Figure 4. Timing Diagram Stretching Z8000 AS and DS

PIN DESCRIPTIONS

Figures 1 and 2, respectively, show the pin functions and assignments of the Z8581.

ADD1, ADD2. *Add Delay 1* (input, active Low) and *Add Delay 2* (input, active Low). These signals control the addition of one, two, or three delay periods to a selected half-cycle of the ZCLK output.

C0, C1. *ZCLK Count 0* (output, active High) and *ZCLK Count 1* (output, active High). These signals indicate, in binary, the number or rising edges of ZCLK that have occurred after the assertion of the STRT input.

INH. *Inhibit Delay* (input, active Low). When asserted, this signal inhibits the functions of inputs ADD1 and ADD2.

OSC. *Time Base Clock* (output, active High). This signal provides a TTL-compatible clock output at the same frequency as the system clock frequency source.

RSTI. *Reset In* (input, active Low). When asserted, this signal indicates a reset condition and initiates the assertion of RSTO synchronized with ZCLK.

RSTO. *Reset Out* (output, active Low). When asserted, this signal indicates that a system reset condition is required, either by RSTI going Low or by a system power-up condition.

STRT. *Start Count* (input, active Low). When asserted, this signal resets a two-bit binary counter and then enables the counter to count the rising edges of the ZCLK output.

STRH. *Delay ZCLK* (input, active Low). When asserted, this signal causes the current half-cycle of the ZCLK output to be delayed (stretched) for as long as STRH is held Low. This control input overrides the ADD1, ADD2, and INH functions.

TCLK. *General-Purpose Clock* (output, MOS-compatible, active High). This signal is the timing output of the general-purpose oscillator. TCLK's frequency is half that of the external oscillator used to drive the general-purpose oscillator.

XTAL1A, XTAL1B. *System Clock Frequency Source A* (input, active High) and *System Clock Frequency Source B* (output, active High). These signals are used by the external oscillator to drive the internal system clock oscillator and the OSC output.

XTAL2A, XTAL2B. *General-Purpose Clock Frequency Source A* (input, active High) and *General-Purpose Clock Frequency Source B* (output, active High). These signals are used by the external oscillator to drive the internal general-purpose clock oscillator.

ZCLK. *System Clock* (output, MOS-compatible, active High). This signal is the timing output of the system clock oscillator. This clock can be modified by the delay (stretch) control inputs. Its frequency, when unmodified, is half that of the external system clock frequency source.

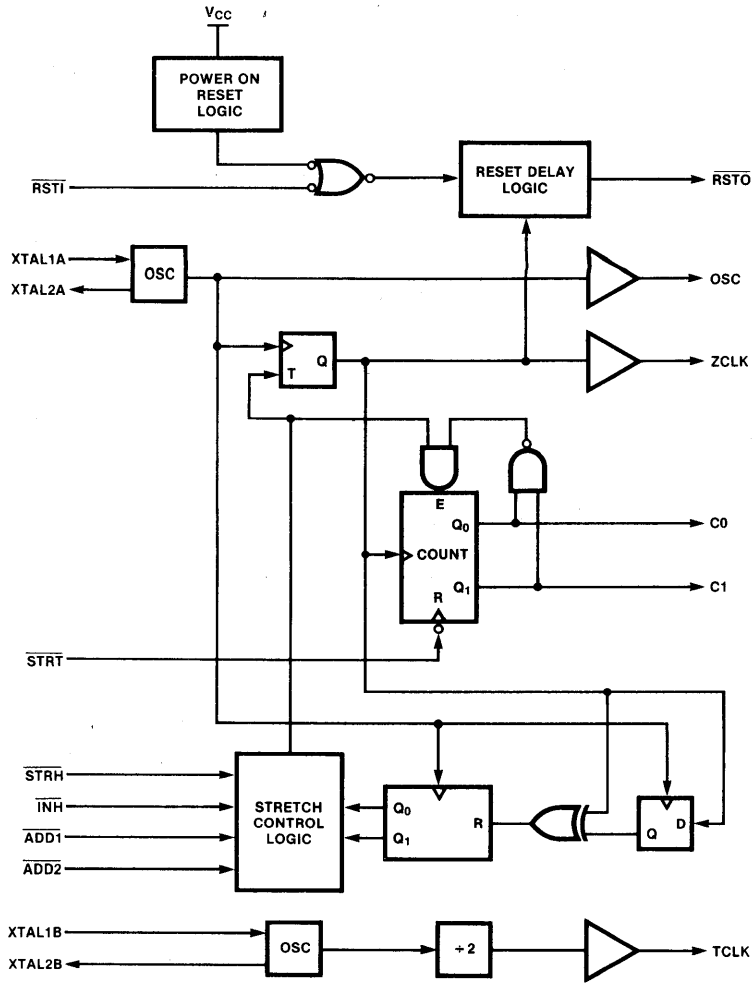


Figure 5. Z8581 Functional Block Diagram

SYSTEM INTERFACE CONSIDERATIONS

Due to the fast rise and fall times produced by the Z8581, transmission line concepts must be applied in order to avoid ringing and reflections on the clock outputs. More specifically, the interconnections between the clock outputs and the loads they are driving must be treated as transmission lines, and it is necessary to match the source impedance of the clock outputs to the characteristic impedances of these transmission lines. In most cases the impedances can be matched by placing termination resistors in series with the clock outputs. These resistors range in value from 22 to 220 ohms, with the value chosen to optimize the clock risetime at the load. (See example below.) It is important to control the impedance seen by the clock output by keeping leads short and avoiding stray inductances wherever possible.

Another important consideration is the bypass capacitor. To avoid distortion of the power supply, the Z8581 requires a high frequency 0.01 μF ceramic capacitor between V_{CC} and ground, and the leads connecting this capacitor to the pins should be kept as short as possible.

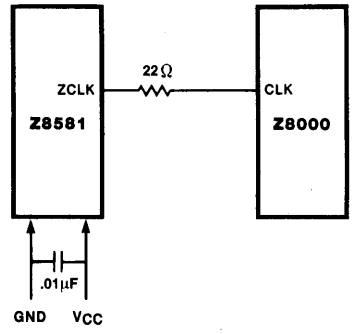
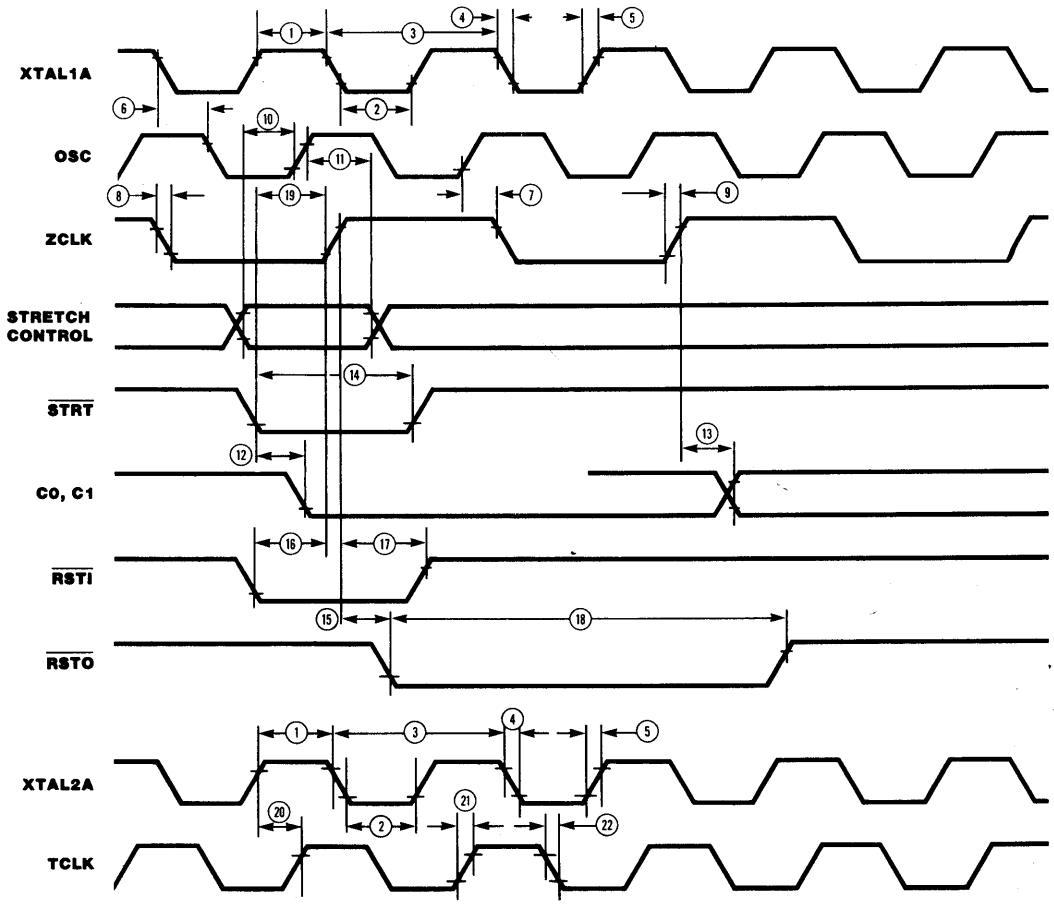


Figure 6. Z8581/Z8000 Interface

AC CHARACTERISTICS†

No.	Symbol	Parameter	6 MHz Min.	6 MHz Max.	10 MHz Min	10 MHz Max	Units
1	TwCH	Clock Input High Width*	31		18		ns
2	TwCL	Clock Input Low Width*	31		18		ns
3	TpC	Clock Input Cycle Time*	82		50		ns
4	TfC	Clock Input Fall Time*		10		7	ns
5	TrC	Clock Input Rise Time*		10		7	ns
6	TdOSC	Clock Input to OSC Delay		30		20	ns
7	TdZC	OSC to ZCLK Delay		10		10	ns
8	TfZC	ZCLK Fall Time		10		10	ns
9	TrZC	ZCLK Rise Time		10		10	ns
10	TsSC	Stretch Controls to OSC ↑ Setup	35		20		ns
11	ThSC	Stretch Controls to OSC ↑ Hold	20		10		ns
12	Td(ST/CR)	$\overline{STRT}$ ↓ to 2-bit Counter Reset Delay		35		25	ns
13	Td(ZC/CC)	ZCLK ↑ to 2-bit Counter Change Delay	−5	10	−5	10	ns
14	Tw(STRT)	$\overline{STRT}$ Low Width	50		30		ns
15	Td(RSTO)	ZCLK ↑ to $\overline{RSTO}$ ↓ Delay		30		20	ns
16	Ts(RSTI)	$\overline{RSTI}$ ↓ to ZCLK ↑ Setup	30		20		ns
17	Th(RSTI)	$\overline{RSTI}$ ↓ to ZCLK ↑ Hold	30		20		ns
18	Tw(RSTO)	$\overline{RSTO}$ Low Width	16		16		cycles
19	Ts(ST/ZC)	$\overline{STRT}$ ↓ to ZCLK↑ Setup to include ZCLK edge	40		30		ns
20	TdTC	Clock Input to TCLK Delay		40		30	ns
21	TrTC	TCLK Rise Time		10		10	ns
22	TfTC	TCLK Fall Time		10		10	ns

*Clock input other than a crystal oscillator
†All timings are preliminary and subject to change.



Timing measurements are made at the following voltages:

	High	Low
ZCLK, TCLK	4.0 V	0.8 V
Output	2.0 V	0.8 V
Input	2.0 V	0.8 V

ABSOLUTE MAXIMUM RATINGS

Voltages on all inputs and outputs with respect to GND . . . . . −0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See ordering information

Storage Temperature . . . . . −65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only: operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

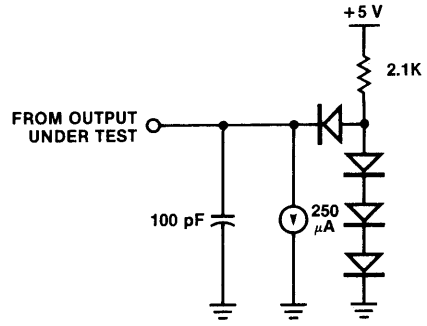
STANDARD TEST CONDITIONS

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND (0 V). Positive current flows into the referenced pin. Available operating temperature ranges are:

- S* = 0°C to +70°C, +4.75 V ≤ V_{CC} ≤ +5.25 V
- E* = −40°C to +85°C, +4.5 V ≤ V_{CC} ≤ +5.25 V
- M* = −55°C to +125°C, +4.5 V ≤ V_{CC} ≤ +5.5 V

*See Ordering Information section for package temperature range and product number.

All ac parameters assume a total load capacitance (including parasitic capacitances) of 100 pF max, except for parameters 8, 9, 21, and 22 (200 pF max). Timing references between two output signals assume a load difference of 50 pF max.



DC CHARACTERISTICS

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	V_{CC}−0.4	V_{CC}+0.3	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	−0.3	0.45	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}+0.3	V	
$V_{IH\ RESET}$	Input High Voltage on RESET pin	2.4	V_{CC} +0.3	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = −250 μA
V_{OH} (ZCLK, TCLK)	Output High Voltage	V_{CC}−0.2		V	I_{OH} = −250 μA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage		±10	μA	0.4 ≤ V_{IN} ≤ +2.4 V
I_{CC}	V_{CC} Supply Current		150	mA	

ORDERING INFORMATION

Product Number	Package/ Temp	Speed	Description
Z8581	C/S	6 MHz	CGC (18-pin)
Z8581	D/S	6 MHz	Same as above
Z8581	P/S	6 MHz	Same as above

Product Number	Package/ Temp	Speed	Description
Z8581	C/S	10 MHz	CGC (18-pin)
Z8581	D/S	10 MHz	Same as above
Z8581	P/S	10 MHz	Same as above

C = Ceramic, D = Cerdip, P = Plastic, S = Standard 0°–70°C

Z8

Family

Zilog

Pioneering the Microworld

Zilog Z8® Family

The New Standard For Single-Chip Microcomputers

September 1983

In 1976, Zilog emerged into the microprocessor world with its Z80 CPU Family. With this classic of 8-bit architecture that has become industry-standard, Zilog established its design expertise and cost-efficient production capability.

While the Z80 earned and maintained strong customer support, inevitable demands for new applications—intelligent terminals, dedicated control and communication—spawned an accompanying need for a new, lean technology. With an instinct for simplicity and elegance, Zilog architects created a microcomputer with the most sophisticated computing power available on a single chip: the Z8 Family. In a bold departure from the standard A/B accumulator tradition, a fresh, register-oriented architecture was devised that challenges the "multi-chip solution." Z8-based designs minimize chip-count while offering a configuration that can be easily expanded to meet the requirements of enhancement options and future improvements.

The Z8 Family encompasses the whole spectrum of system development, from prototyping to full production. For prototyping and preproduction, or where code flexibility is important, the Z8603 and Z8613 Protopack EPROM-based parts are the most appropriate. The ROM-based Z8601 and Z8611 microcomputers are used in high-volume production applications after the software has been perfected. The Z8603 is pin-compatible with the Z8601, and the Z8613 is compatible with the Z8611.

For ROMless applications, two versions of the Z8 microcomputer are available: the 40-pin Z8681 and Z8682 ROMless microcomputers. In addition, there is a military version of the Z8611 4K ROM device, available in both 40-pin ceramic and 44-pin leadless chip carrier packages.

The Z8671 MCU is a complete microcomputer pre-programmed with a BASIC/Debug interpreter. This device, operating with both external ROM or RAM and on-chip memory registers, is ideal for most industrial control applications, or whenever fast and efficient program development is necessary.

Dedicated control is the key word for Z8 applications. Since speed is a prime consideration in such applications, the entire Z8 family is available in both 8 and 12 MHz versions, supported by either of two development tools: the Z8 DM Development Module and the Z-SCAN 8. The DM provides elementary ICE capability, and the Z-SCAN 8 module provides full in-circuit emulation (ICE) capability including trace memory. With these tools, the user is equipped for practically any type of Z8 microcomputer development.

Z8 Family of Products

Product	Part Number	RAM	ROM Capacity (Bytes)	UART	Programmable I/O Pins (max.)	Dedicated I/O Pins	PCB Footprint	Comments
2K ROM	Z8601	128	2K	H	32, 4 ports	8 Power, Control	40-Pin	Masked ROM part, used primarily for high-volume production.
2K Protopack	Z8603	128	0	H	32, 4 ports	8 Power, Control plus 24 EPROM	40-Pin	Piggyback part used where program flexibility is required (prototyping).
4K ROM	Z8611	128	4K	H	32, 4 ports	8 Power, Control	40-Pin	Masked ROM part, used primarily for high-volume production.
4K Development part	Z8612	128	0	H	32, 4 ports	8 Power, Control plus 24 external memory	64-Pin	ROMless part used primarily in development systems.
4K Protopack	Z8613	128	0	H	32, 4 ports	8 Power, Control plus 24 EPROM	40-Pin	Piggyback part used where program flexibility is required (prototyping).
BASIC/ Debug	Z8671	128	2K	H	24, 3 ports	8 Power, Control	40-Pin	BASIC/Debug part used in low-volume applications.
ROMless	Z8681/ Z8682	128	0	H	24, 3 ports	8 Power, Control plus 8 external memory	40-Pin	Low-cost ROMless production part with reduced I/O. Program memory is external.
Z8-UPC	Z8090	256	2K	S	24, 3 ports	8 Power, Control plus 8 master bus	40-Pin	Masked ROM part, master CPU bus is Z-BUS compatible.
Z8-UPC	Z8590	256	2K	S	24, 3 ports	8 Power, Control plus 8 master bus	40-Pin	Masked ROM part, master CPU bus is general-purpose.
Z8-UPC	Z8094	256	0	S	24, 3 ports	8 Power, Control plus 8 master CPU 24, RAM/ PROM	40-Pin	Piggyback part used where program flexibility is needed. Prototypes, Z-BUS compatible.
Z8-UPC	Z8594	256	0	S	24, 3 ports	8 Power, Control plus 8 master CPU 24, RAM/ PROM	40-Pin	Piggyback part used where program flexibility is needed. Prototypes, general-purpose.

H = Hardware
S = Software

Z8® Family of Microcomputers Z8601 • Z8603

Zilog

Product Specification

September 1983

Z8601 Single-Chip Microcomputer with 2K ROM
Z8603 Prototyping Device with EPROM Interface

Features

- Complete microcomputer, 2K bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 62K bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Average instruction execution time of 1.5 μs, maximum of 3 μs.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working register groups in 1 μs.
- On-chip oscillator that accepts crystal or external clock drive.
- Low-power standby option which retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL-compatible.

General Description

The Z8601 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8601 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8601 can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 2K bytes of internal ROM, a traditional microprocessor that manages up to 124K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

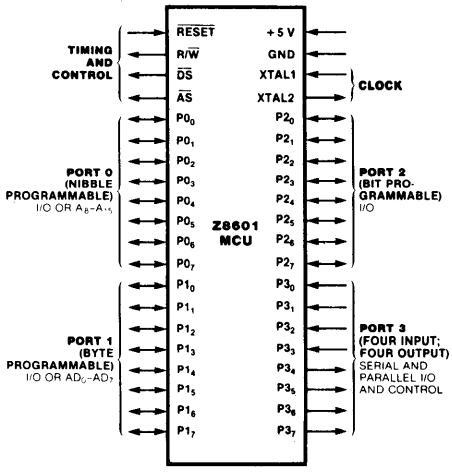


Figure 1. Pin Functions

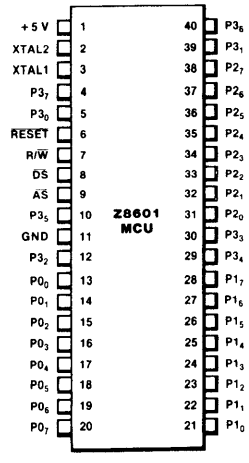


Figure 2. Pin Assignments

Architecture

Z8601 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8601 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8601 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 124K bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

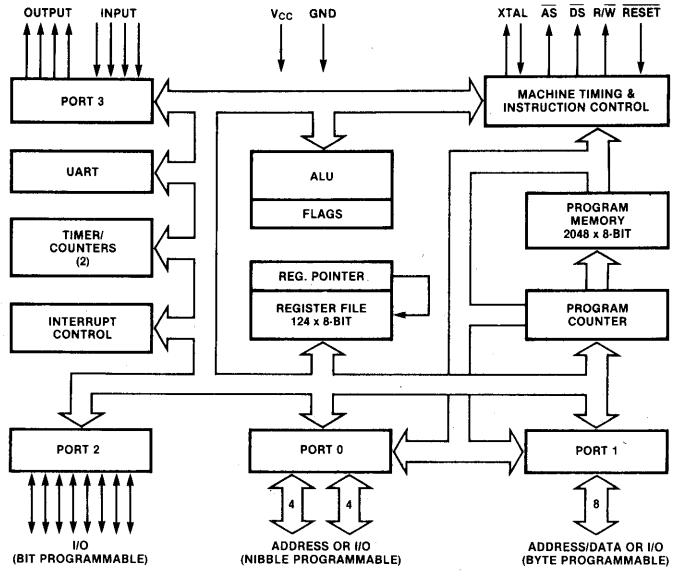


Figure 3. Functional Block Diagram

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8601. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8601 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.

Address Spaces

Program Memory. The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 2048 bytes consist of on-chip mask-programmed ROM. At addresses 2048 and greater, the Z8601 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

Data Memory. The Z8601 can address 62K bytes of external data memory beginning at location 2048 (Figure 5). External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 144-byte register file includes four I/O port registers (R0–R3), 124 general-purpose registers (R4–R127) and 16 control and status registers (R240–R255). These registers are assigned the address locations shown in Figure 6.

Z8601 instructions can access registers

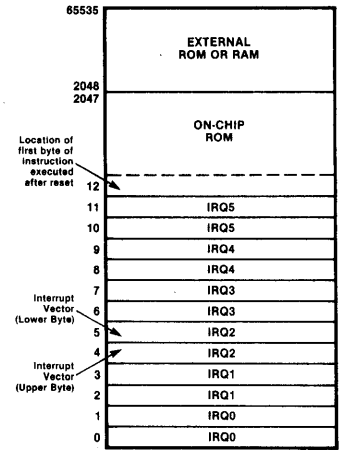


Figure 4. Program Memory Map

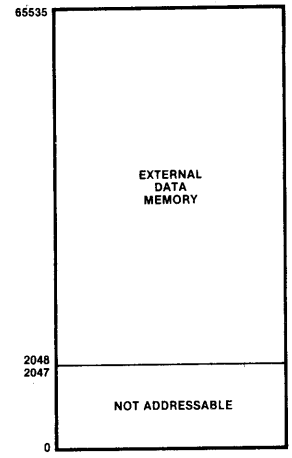
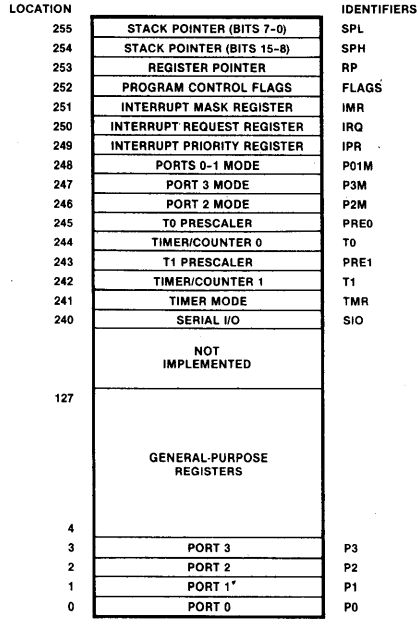


Figure 5. Data Memory Map



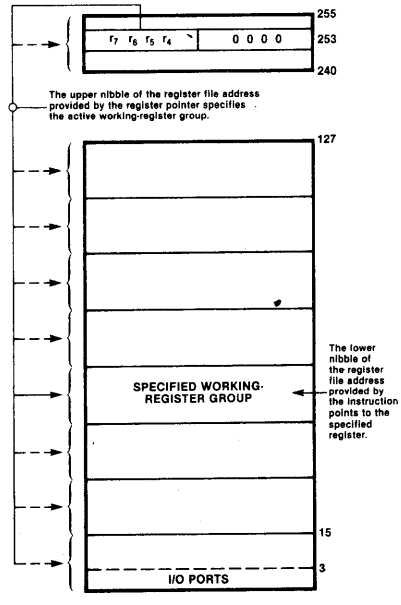


Figure 7. The Register Pointer

Address Spaces (Continued)

directly or indirectly with an 8-bit address field. The Z8601 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 2048 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4–R127).

Serial Input/ Output

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second for 8 MHz and 94.8K bits/second for 12 MHz.

The Z8601 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.

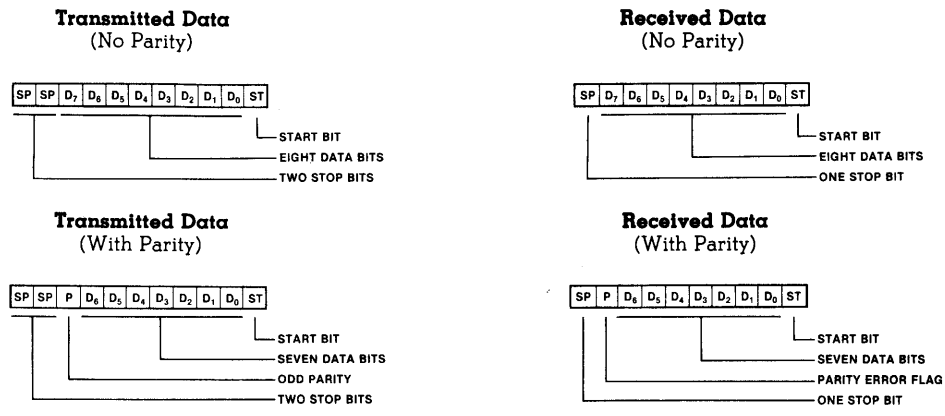


Figure 8. Serial Data Formats

Counter/ Timers

The Z8601 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock (4 MHz maximum for the 8 MHz device and 6 MHz maximum for the 12 MHz device.) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O Ports

The Z8601 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY_1 and $\overline{DAV}_1$ (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8601 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input and $P3_4$ as a Bus Request output.

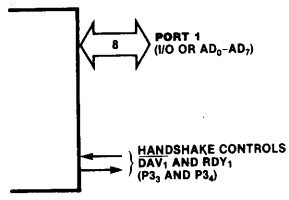


Figure 9a. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV}_0$ and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

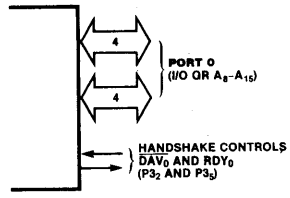


Figure 9b. Port 0

Port 2 bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

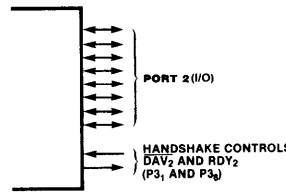


Figure 9c. Port 2

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0-IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

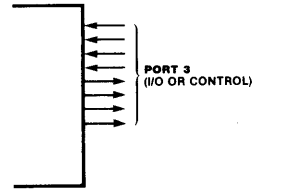


Figure 9d. Port 3

Interrupts

The Z8601 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$–$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8601 interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors ($C_1 \leq 15$ pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental type, 8/12 MHz maximum
- Series resistance, $R_s \leq 100\ \Omega$

Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the V_{MM} (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.

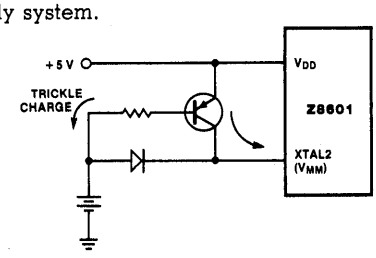


Figure 10. Recommended Driver Circuit for Power Down Operation

Z8603 Protopack Emulator

The Z8603 MPE (Protopack) is used for prototype development and preproduction of mask-programmed applications. The Protopack is a ROMless version of the standard Z8601, housed in a pin-compatible 40-pin package (Figure 11).

To provide pin compatibility and interchangeability with the standard mask-programmed device, the Protopack carries (piggy-backs) a 24-pin socket for a direct interface to program memory (Figure 1). The 24-pin socket is equipped with 11 ROM address lines, 8 ROM data lines and necessary control lines for interface to 2716 EPROM for the first 2K bytes of program memory.

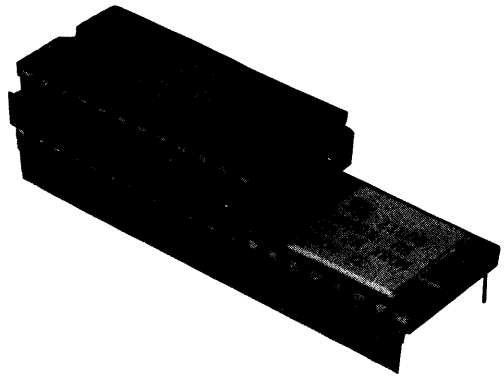


Figure 11. The Z8603 Microcomputer Protopack Emulator

Pin compatibility allows the user to design the pc board for a final 40-pin mask-programmed Z8601, and, at the same time, allows the use of the Protopack to build the prototype and pilot production units. When the final program is established, the user can then switch over to the 40-pin mask-programmed Z8601 for large volume production. The Protopack is also useful in small volume applica tions where masked ROM setup time, mask charges, etc., are prohibitive and program flexibility is desired.

Compared to the conventional EPROM versions of the single-chip microcomputers, the Protopack approach offers two main advantages:

- Ease of developing various programs during the prototyping stage. For instance, in applications where the same hardware configuration is used with more than one program, the Z8603 Protopack allows economical program storage in separate EPROMs (or PROMs), whereas the use of separate EPROM-based single-chip microcomputers is more costly.
- Elimination of long lead time in procuring EPROM-based microcomputers.

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats

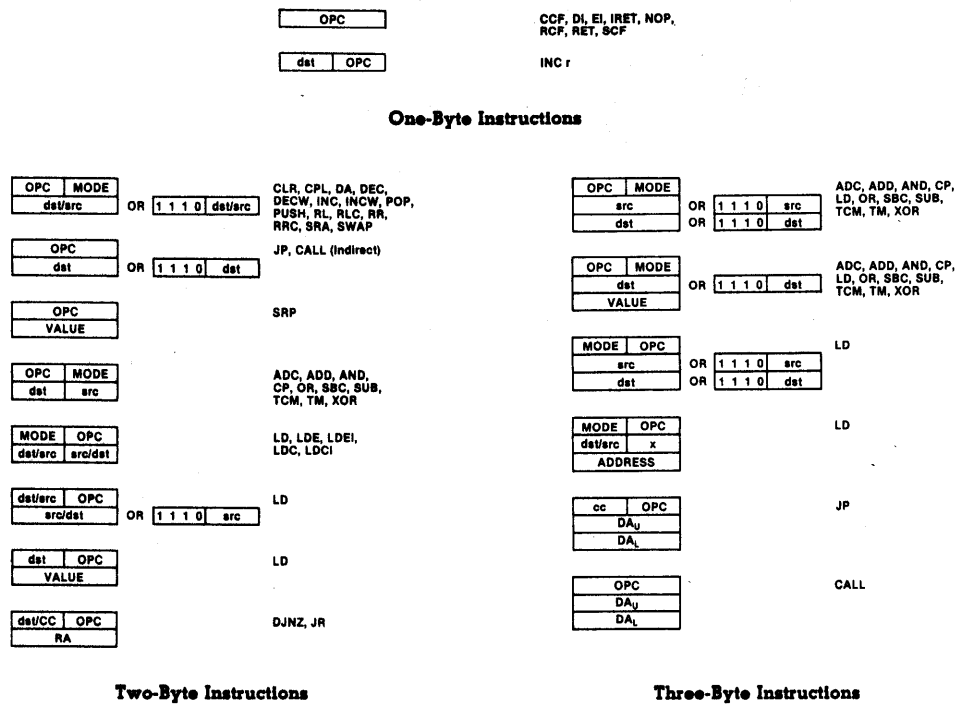


Figure 12. Instruction Formats

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR(7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R IR Im IR IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst	R IR		90 91	* * * * - -
RLC dst	R IR		10 11	* * * * - -
RR dst	R IR		E0 E1	* * * * - -
RRC dst	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

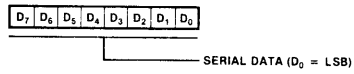
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

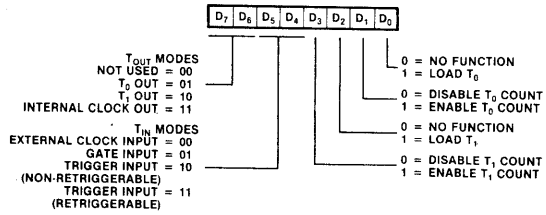
For example, to determine the opcode of an ADC instruction use the addressing modes r (destination) and Ir (source). The result is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

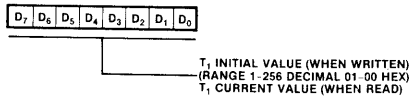
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



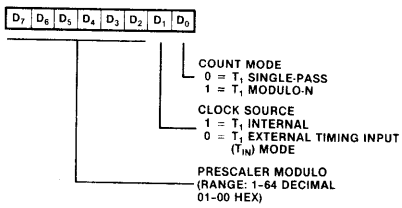
R241 TMR
Timer Mode Register
($F1_H$; Read/Write)



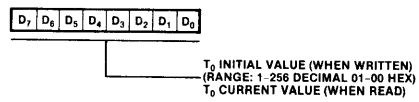
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



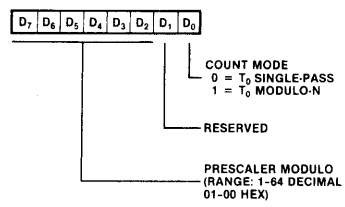
R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



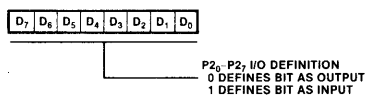
R244 T0
Counter/Timer 0 Register
($F4_H$; Read/Write)



R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

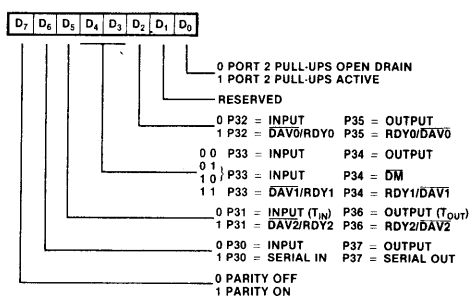
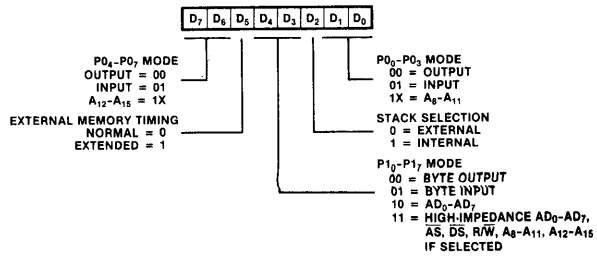
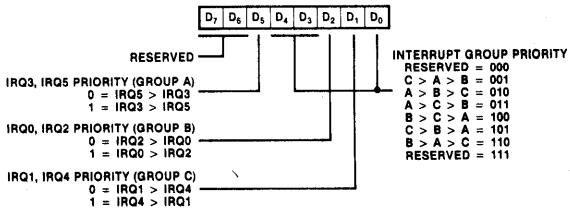


Figure 13. Control Registers

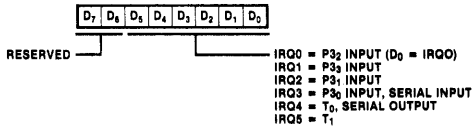
R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



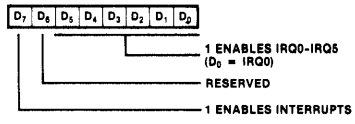
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



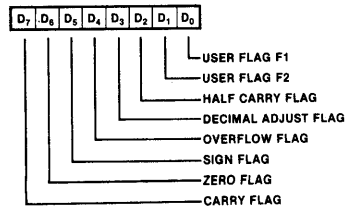
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



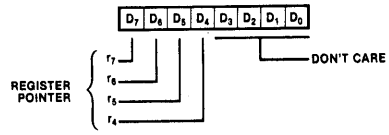
R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



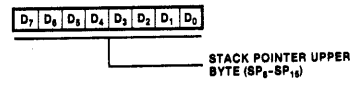
R252 FLAGS
Flag Register
(FC_H; Read/Write)



R253 RP
Register Pointer
(FD_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

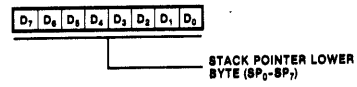
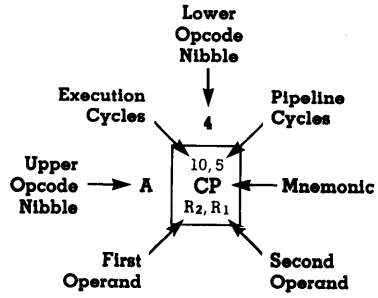


Figure 13. Control Registers

Z8601 Opcode Map

Upper Nibble (Hex)	Lower Nibble (Hex) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R1	6,5 **DEC** IR1	6,5 **ADD** r1, r2	6,5 **ADD** r1, Ir2	10,5 **ADD** R2, R1	10,5 **ADD** IR2, R1	10,5 **ADD** R1, IM	10,5 **ADD** IR1, IM	6,5 **LD** r1, R2	6,5 **LD** r2, R1	12/10,5 **DJNZ** r1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r1	
1	6,5 **RLC** R1	6,5 **RLC** IR1	6,5 **ADC** r1, r2	6,5 **ADC** r1, Ir2	10,5 **ADC** R2, R1	10,5 **ADC** IR2, R1	10,5 **ADC** R1, IM	10,5 **ADC** IR1, IM								
2	6,5 **INC** R1	6,5 **INC** IR1	6,5 **SUB** r1, r2	6,5 **SUB** r1, Ir2	10,5 **SUB** R2, R1	10,5 **SUB** IR2, R1	10,5 **SUB** R1, IM	10,5 **SUB** IR1, IM								
3	8,0 **JP** IRR1	6,1 **SRP** IM	6,5 **SBC** r1, r2	6,5 **SBC** r1, Ir2	10,5 **SBC** R2, R1	10,5 **SBC** IR2, R1	10,5 **SBC** R1, IM	10,5 **SBC** IR1, IM								
4	8,5 **DA** R1	8,5 **DA** IR1	6,5 **OR** r1, r2	6,5 **OR** r1, Ir2	10,5 **OR** R2, R1	10,5 **OR** IR2, R1	10,5 **OR** R1, IM	10,5 **OR** IR1, IM								
5	10,5 **POP** R1	10,5 **POP** IR1	6,5 **AND** r1, r2	6,5 **AND** r1, Ir2	10,5 **AND** R2, R1	10,5 **AND** IR2, R1	10,5 **AND** R1, IM	10,5 **AND** IR1, IM								
6	6,5 **COM** R1	6,5 **COM** IR1	6,5 **TCM** r1, r2	6,5 **TCM** r1, Ir2	10,5 **TCM** R2, R1	10,5 **TCM** IR2, R1	10,5 **TCM** R1, IM	10,5 **TCM** IR1, IM								
7	10/12,1 **PUSH** R2	12/14,1 **PUSH** IR2	6,5 **TM** r1, r2	6,5 **TM** r1, Ir2	10,5 **TM** R2, R1	10,5 **TM** IR2, R1	10,5 **TM** R1, IM	10,5 **TM** IR1, IM								
8	10,5 **DECW** RR1	10,5 **DECW** IR1	12,0 **LDE** r1, Irr2	18,0 **LDEI** Ir1, Irr2												6,1 **DI**
9	6,5 **RL** R1	6,5 **RL** IR1	12,0 **LDE** r2, Irr1	18,0 **LDEI** Ir2, Irr1												6,1 **EI**
A	10,5 **INCW** RR1	10,5 **INCW** IR1	6,5 **CP** r1, r2	6,5 **CP** r1, Ir2	10,5 **CP** R2, R1	10,5 **CP** IR2, R1	10,5 **CP** R1, IM	10,5 **CP** IR1, IM								14,0 **RET**
B	6,5 **CLR** R1	6,5 **CLR** IR1	6,5 **XOR** r1, r2	6,5 **XOR** r1, Ir2	10,5 **XOR** R2, R1	10,5 **XOR** IR2, R1	10,5 **XOR** R1, IM	10,5 **XOR** IR1, IM								16,0 **IRET**
C	6,5 **RRC** R1	6,5 **RRC** IR1	12,0 **LDC** r1, Irr2	18,0 **LDCI** Ir1, Irr2				10,5 **LD** r1, x, R2								6,5 **RCF**
D	6,5 **SRA** R1	6,5 **SRA** IR1	12,0 **LDC** r2, Irr1	18,0 **LDCI** Ir2, Irr1	20,0 **CALL*** IRR1		20,0 **CALL** DA	10,5 **LD** r2, x, R1								6,5 **SCF**
E	6,5 **RR** R1	6,5 **RR** IR1		6,5 **LD** r1, Ir2	10,5 **LD** R2, R1	10,5 **LD** IR2, R1	10,5 **LD** R1, IM	10,5 **LD** IR1, IM								6,5 **CCF**
F	8,5 **SWAP** R1	8,5 **SWAP** IR1		6,5 **LD** Ir1, r2		10,5 **LD** R2, IR1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R1 or r1 = Dst Address
R2 or r2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- □ $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- □ GND = 0 V
- □ $0°C \leq T_A \leq +70°C$*

*See Ordering Information section for package temperature range and product number.

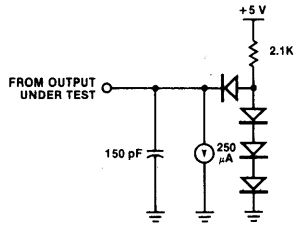


Figure 14. Test Load 1

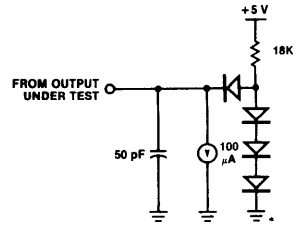


Figure 15. Test Load 2

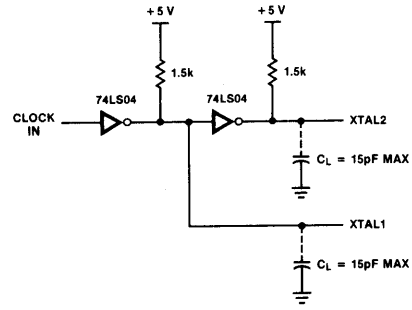


Figure 16. TTL External Clock Interface Circuit (Both the clock and its complement are required)

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250\ \mu A$
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = +2.0$ mA
I_{IL}	Input Leakage	-10	10	µA	$0\ V \leq V_{IN} \leq +5.25\ V$
I_{OL}	Output Leakage	-10	10	µA	$0\ V \leq V_{IN} \leq +5.25\ V$
I_{IR}	Reset Input Current		-50	µA	$V_{CC} = +5.25\ V$, $V_{RL} = 0\ V$
I_{CC}	V_{CC} Supply Current		180	mA	
I_{MM}	V_{MM} Supply Current		10	mA	Power Down Mode
V_{MM}	Backup Supply Voltage	3	V_{CC}	V	Power Down

External I/O or Memory Read and Write Timing

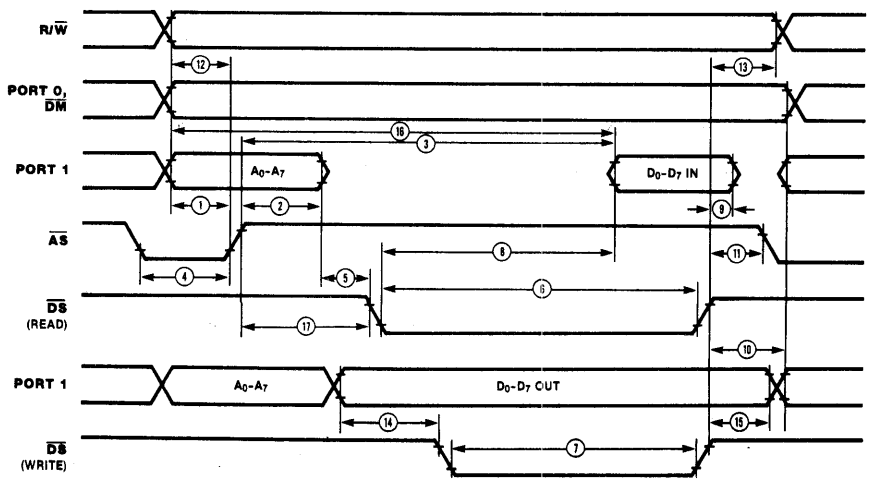


Figure 17. External I/O or Memory Read/Write

No.	Symbol	Parameter	Z8601/3 Min	Z8601/3 Max	Z8601-12 Min	Z8601-12 Max	Notes*†
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		55		1,2,3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/$\overline{W}$ Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		1,2,3

NOTES:
1. Test Load 1
2. Timing numbers given are for minimum TpC.
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 TpC.
5. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* All units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Additional Timing Table

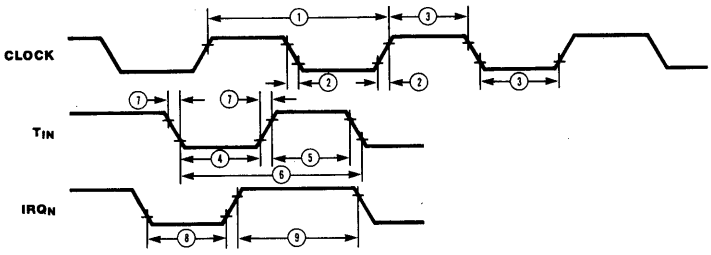


Figure 18. Additional Timing

No.	Symbol	Parameter	Z8601/3 Min	Z8601/3 Max	Z8601-12 Min	Z8601-12 Max	Notes*†
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin,TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIL	Interrupt Request Input Low Time	3TpC		3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:
1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".
3. Interrupt request via Port 3 ($P3_1$-$P3_3$).
4. Interrupt request via Port 3 ($P3_0$).

* Units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Z8603 Memory Port Timing

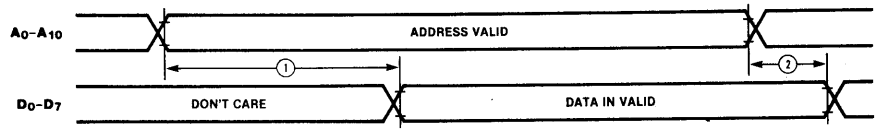


Figure 19. Memory Port Timing

No.	Symbol	Parameter	Z8601 Min	Z8601 Max	Z8601 Min	Z8601 Max	Notes*
1	TdA(DI)	Address Valid to Data Input Delay		460		320	1,2
2	ThDI(A)	Data In Hold Time	0		0		1

NOTES:
1. Test Load 2
2. This is a Clock-Cycle-Dependent parameter. For clock frequencies other than the maximum, use the following formula:
   Z8601/3 = 5 TpC − 165
   Z8601/3-12 = 5 TpC − 95

* Units are nanoseconds unless otherwise specified; timings are preliminary and subject to change.

Handshake Timing

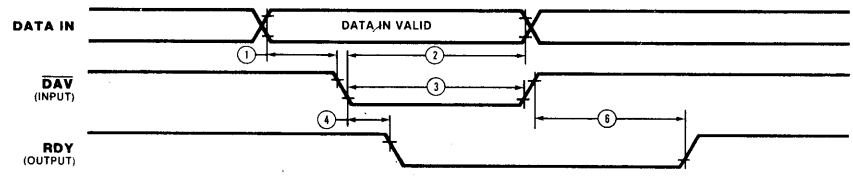


Figure 20a. Input Handshake

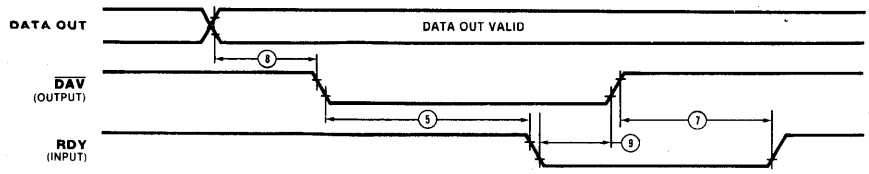


Figure 20b. Output Handshake

No.	Symbol	Parameter	Z8601/3 Min	Z8601/3 Max	Z8601/3-12 Min	Z8601/3-12 Max	Notes*†
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAV0rRDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake
4. All timing regerences use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* Units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8601/3 Equation	Z8601/3-12 Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

* Add 2TpC when using extended memory timing

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8601	CE	8.0 MHz	Z8 MCU (2K ROM, 40-pin)
Z8601	CS	8.0 MHz	Same as above
Z8601	PE	8.0 MHz	Same as above
Z8601	PS	8.0 MHz	Same as above
Z8601-12	CE	12.0 MHz	Z8 MCU (2K ROM, 40-pin)
Z8601-12	CS	12.0 MHz	Same as above
Z8601-12	PE	12.0 MHz	Z8 MCU (2K ROM, 40-pin)
Z8601-12	PS	12.0 MHz	Same as above
Z8603	RS	8.0 MHz	Z8 MCU (2K XROM, Prototyping Device, 40-pin)
Z8603-12	RS	12.0 MHz	Same as above

NOTES: C = Ceramic, P = Plastic, R = Protopack; E = 40° to 85°C, S = 0°C to +70°C.

Z8® Family of Microcomputers Z8611 • Z8612 • Z8613

Product Specification

Zilog

September 1983

Z8611 Single-Chip Microcomputer with 4K ROM
Z8612 Development Device with Memory Interface
Z8613 Prototyping Device with EPROM Interface

Features

- Complete microcomputer, 4K bytes of ROM, 128 bytes of RAM, 32 I/O lines, and up to 60K bytes addressable external space each for program and data memory.
- 144-byte register file, including 124 general-purpose registers, four I/O port registers, and 16 status and control registers.
- Average instruction execution time of 1.5 μs, maximum of 3 μs.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any of nine working-register groups in 1 μs.
- On-chip oscillator which accepts crystal or external clock drive.
- Low-power standby option that retains contents of general-purpose registers.
- Single +5 V power supply—all pins TTL-compatible.

General Description

The Z8611 microcomputer introduces a new level of sophistication to single-chip architecture. Compared to earlier single-chip microcomputers, the Z8611 offers faster execution; more efficient use of memory; more sophisticated interrupt, input/output and bit-manipulation capabilities; and easier system expansion.

Under program control, the Z8611 can be tailored to the needs of its user. It can be configured as a stand-alone microcomputer with 4K bytes of internal ROM, a traditional microprocessor that manages up to 120K bytes of external memory, or a parallel-processing element in a system with other processors and peripheral controllers linked by the Z-BUS. In all configurations, a large number of pins remain available for I/O.

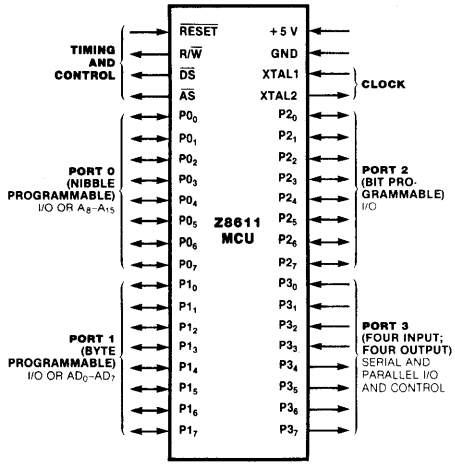


Figure 1. Z8611 MCU Pin Functions

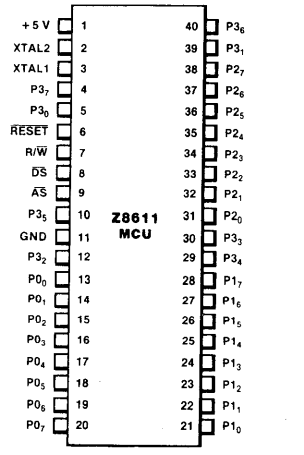


Figure 2. Z8611 MCU Pin Assignments

Architecture

Z8611 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8611 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8611 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 120K bytes of external memory (Figure 3).

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

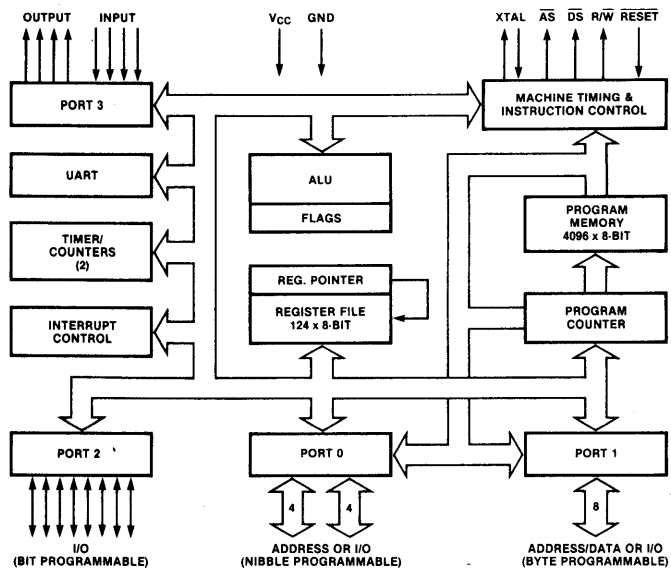


Figure 3. Functional Block Diagram

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8611. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8611 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.

Address Spaces

Program Memory. The 16-bit program counter addresses 64K bytes of program memory space. Program memory can be located in two areas: one internal and the other external (Figure 4). The first 4096 bytes consist of on-chip mask-programmed ROM. At addresses 4096 and greater, the Z8611 executes external program memory fetches.

The first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that correspond to the six available interrupts.

Data Memory. The Z8611 can address 60K bytes of external data memory beginning at location 4096 (Figure 5). External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 144-byte register file includes four I/O port registers (R0–R3), 124 general-purpose registers (R4–R127) and 16 control and status registers (R240–R255). These registers are assigned the address locations shown in Figure 6.

Z8611 instructions can access registers

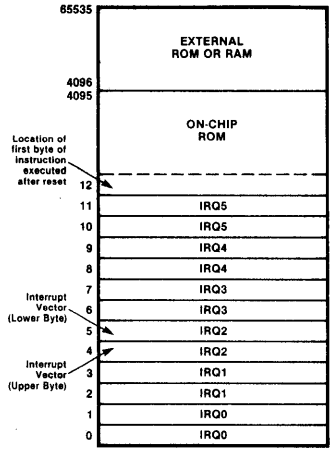


Figure 4. Program Memory Map

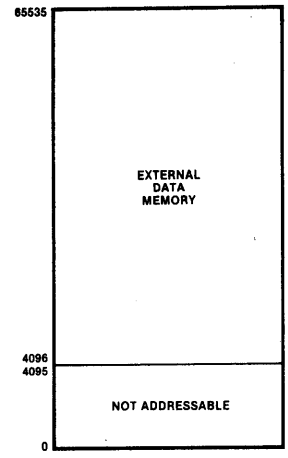


Figure 5. Data Memory Map

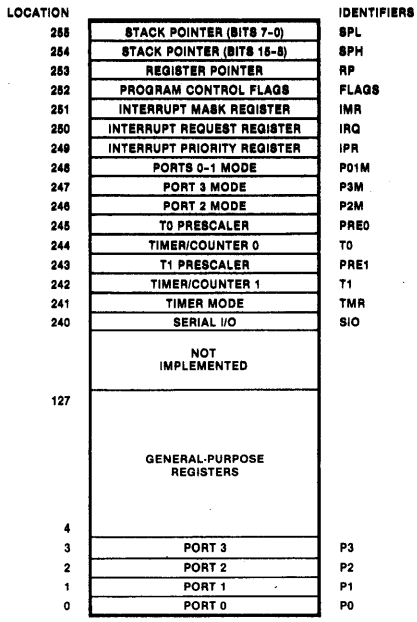


Figure 6. The Register File

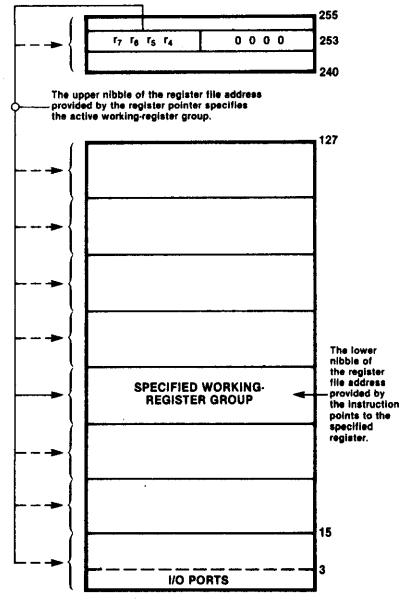


Figure 7. The Register Pointer

Address Spaces (Continued)

directly or indirectly with an 8-bit address field. The Z8611 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 7). The Register Pointer addresses the starting location of the active working-register group.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between locations 4096 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

Serial Input/ Output

Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second for 8 MHz and 94.8K bits/second for 12 MHz.

The Z8611 automatically adds a start bit and two stop bits to transmitted data (Figure 8). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth bit is the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.

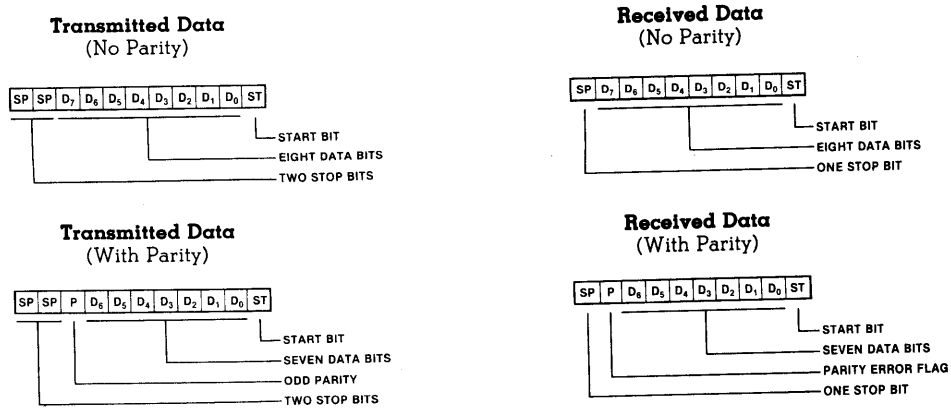


Figure 8. Serial Data Formats

Counter/ Timers

The Z8611 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable and can be the internal microprocessor clock (4 MHz maximum for the 8 MHz device and a 6 MHz maximum for the 12 MHz device.) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock (1 MHz maximum), a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O Ports

The Z8611 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY_1 and $\overline{DAV}_1$ (Ready and Data Available).

Memory locations greater than 4096 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8611 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input, and $P3_4$ as a Bus Request output.

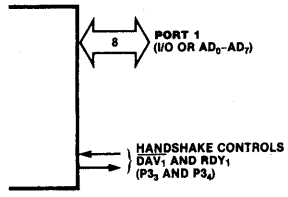


Figure 9a. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV}_0$ and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8–A_{11} (lower nibble) or A_8–A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

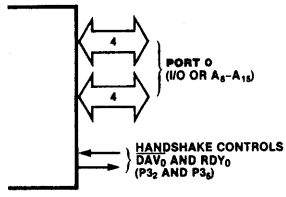


Figure 9b. Port 0

Port 2 bits can be programmed independently as input or output. This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

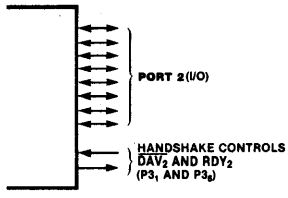


Figure 9c. Port 2

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$–$P3_3$) and four output ($P3_4$–$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0–IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

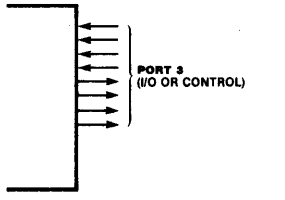


Figure 9d. Port 3

Interrupts

The Z8611 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$-$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8611 interrupts are vectored. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitors ($C_1 \leq 15$ pF) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental type, 8/12 MHz maximum
- Series resistance, $R_s \leq 100\ \Omega$

Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the V_{MM} (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 10 shows the recommended circuit for a battery back-up supply system.

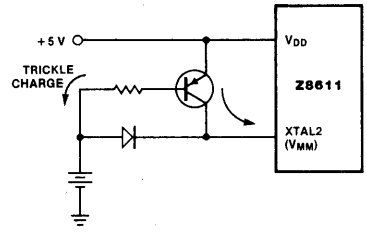


Figure 10. Recommended Driver Circuit for Power Down Operation

Z8612 Development Device

This 64-pin development version of the 40-pin mask-programmed Z8611 (Figure 11) allows the user to prototype the system in hardware with an actual device and to develop the code that is eventually mask-programmed into the on-chip ROM of the Z8611.

The Z8612 is identical to the Z8611 with the following exceptions:

- The internal ROM has been removed.
- The ROM address lines and data lines are buffered and brought out to external pins.
- Control lines for the new memory have been added.

Pin Description. The functions of the Z8612 I/O lines, $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$, XTAL1, XTAL2 and $\overline{RESET}$ are identical to those of their Z8611 counterparts. The functions of the remaining 24 pins are as follows:

A_0-A_{11}. *Program Memory Address* (outputs). A_0-A_{11} access the first 4K bytes of program memory.

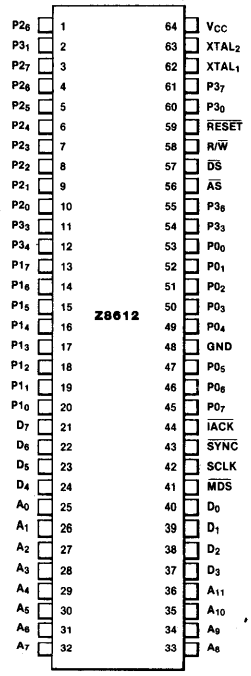


Figure 11. Z8612 Pin Assignments

Z8612 Development Device (Continued)

D_0-D_7. *Program Data* (inputs). Program data from the first 4K bytes of program memory is input through pins D_0-D_7.

IACK. *Interrupt Acknowledge* (output, active High). IACK is driven High in response to an interrupt during the interrupt machine cycle.

$\overline{MDS}$. *Program Memory Data Strobe* (output, active Low). $\overline{MDS}$ is Low during an instruction fetch cycle when the first 4K bytes of program memory are being accessed.

SCLK. *System Clock* (output). SCLK is the internal clock output through a buffer. The clock rate is equal to one-half the crystal frequency.

$\overline{SYNC}$. *Instruction Sync* (output, active Low). This strobe output is forced Low during the internal clock period preceding an opcode fetch.

Z8613 Protopack Emulator

The Z8613 MPE (Protopack) is used for prototype development and preproduction of mask-programmed applications. The Protopack is a ROMless version of the standard Z8611, housed in a pin-compatible 40-pin package (Figure 12).

To provide pin compatibility and interchangeability with the standard mask-programmed device, the Protopack carries (piggy-back) a 24-pin socket for a direct interface to program memory (Figure 1). The 24-pin socket is equipped with 12 ROM address lines, 8 ROM data lines and necessary control lines for interface to 2732 EPROM for the first 4K bytes of program memory.

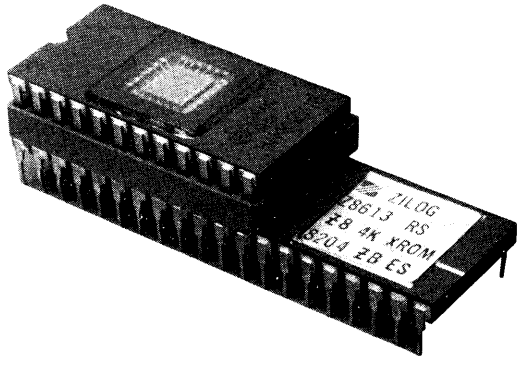


Figure 12. The Z8613 Microcomputer Protopack Emulator

Pin compatibility allows the user to design the pc board for a final 40-pin mask-programmed Z8611, and, at the same time, allows the use of the Protopack to build the prototype and pilot production units. When the final program is established, the user can then switch over to the 40-pin mask-programmed Z8611 for large volume production. The Protopack is also useful in small volume applications where masked ROM setup time, mask charges, etc., are prohibitive and program flexibility is desired.

Compared to the conventional EPROM versions of the single-chip microcomputers, the Protopack approach offers two main advantages:

- Ease of developing various programs during the prototyping stage: For instance, in applications where the same hardware configuration is used with more than one program, the Z8613 Protopack allows economical program storage in separate EPROMs (or PROMs), whereas the use of separate EPROM-based single-chip microcomputers is more costly.
- Elimination of long lead time in procuring EPROM-based microcomputers.

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	*Working-register address only*
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Instruction Set Notation (Continued)

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

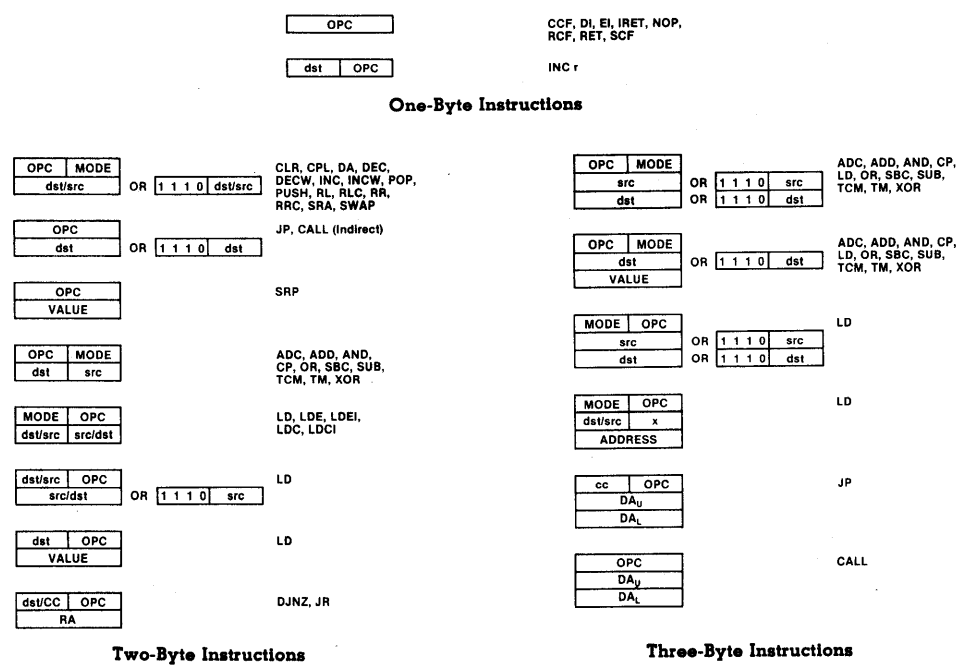
Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats



Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r = 0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r = 0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c = 0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c = 0-F	- - - - - -
LD dst,src dst ← src	r r R r X r r Ir R R R IR IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r = 0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst	R IR		90 91	* * * * - -
RLC dst	R IR		10 11	* * * * - -
RR dst	R IR		E0 E1	* * * * - -
RRC dst	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

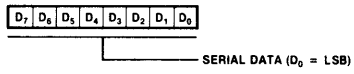
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

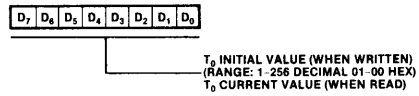
For example, to determine the opcode of an ADC instruction use the addressing modes r (destination) and Ir (source). The result is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

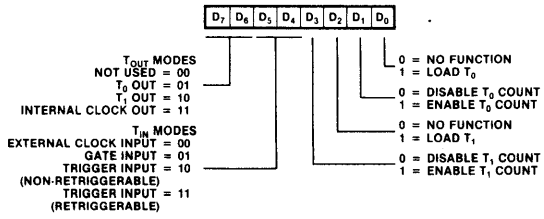
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



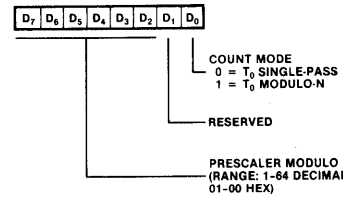
R244 T0
Counter/Timer 0 Register
($F4_H$; Read/Write)



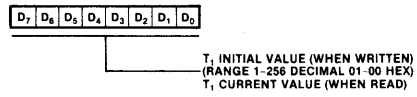
R241 TMR
Timer Mode Register
($F1_H$; Read/Write)



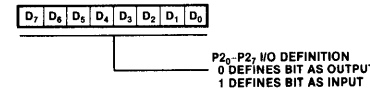
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



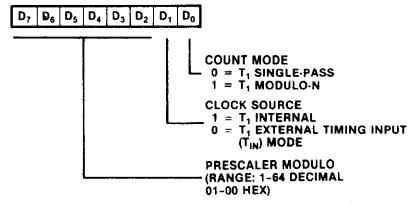
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

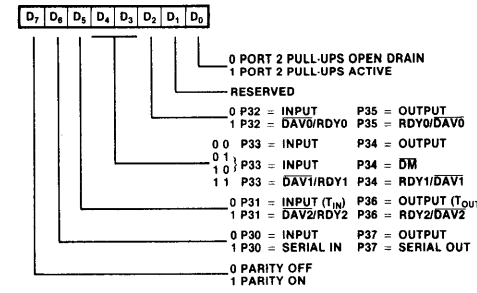
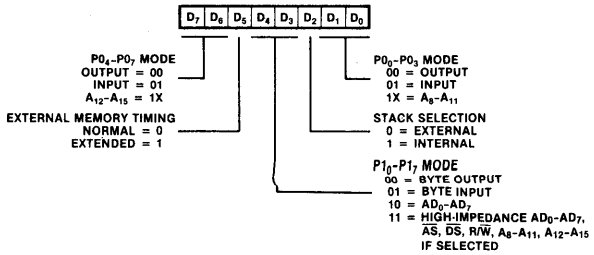
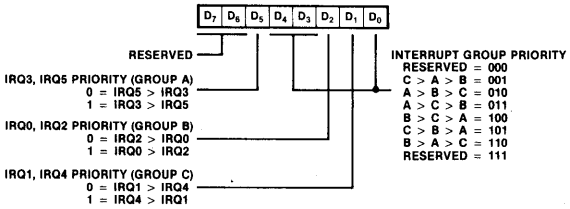


Figure 14. Control Registers

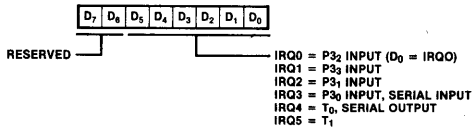
R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



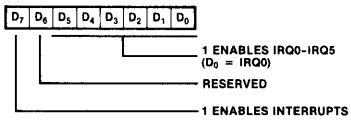
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



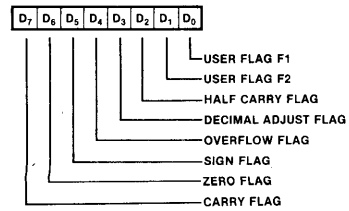
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



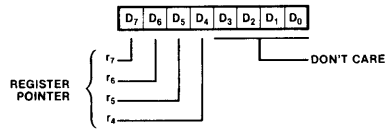
R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



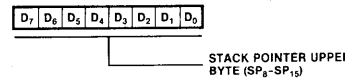
R252 FLAGS
Flag Register
(FC_H; Read/Write)



R253 RP
Register Pointer
(FD_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

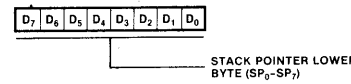
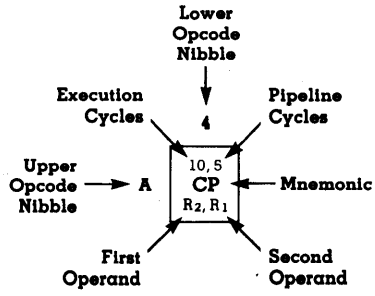


Figure 14. Control Registers

Opcode Map

Upper Nibble (Hex)	Lower Nibble (Hex) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1, r_2	6,5 **ADD** r_1, Ir_2	10,5 **ADD** R_2, R_1	10,5 **ADD** IR_2, R_1	10,5 **ADD** R_1, IM	10,5 **ADD** IR_1, IM	6,5 **LD** r_1, R_2	6,5 **LD** r_2, R_1	12/10,5 **DJNZ** r_1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r_1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r_1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1, r_2	6,5 **ADC** r_1, Ir_2	10,5 **ADC** R_2, R_1	10,5 **ADC** IR_2, R_1	10,5 **ADC** R_1, IM	10,5 **ADC** IR_1, IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1, r_2	6,5 **SUB** r_1, Ir_2	10,5 **SUB** R_2, R_1	10,5 **SUB** IR_2, R_1	10,5 **SUB** R_1, IM	10,5 **SUB** IR_1, IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1, r_2	6,5 **SBC** r_1, Ir_2	10,5 **SBC** R_2, R_1	10,5 **SBC** IR_2, R_1	10,5 **SBC** R_1, IM	10,5 **SBC** IR_1, IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1, r_2	6,5 **OR** r_1, Ir_2	10,5 **OR** R_2, R_1	10,5 **OR** IR_2, R_1	10,5 **OR** R_1, IM	10,5 **OR** IR_1, IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1, r_2	6,5 **AND** r_1, Ir_2	10,5 **AND** R_2, R_1	10,5 **AND** IR_2, R_1	10,5 **AND** R_1, IM	10,5 **AND** IR_1, IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1, r_2	6,5 **TCM** r_1, Ir_2	10,5 **TCM** R_2, R_1	10,5 **TCM** IR_2, R_1	10,5 **TCM** R_1, IM	10,5 **TCM** IR_1, IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1, r_2	6,5 **TM** r_1, Ir_2	10,5 **TM** R_2, R_1	10,5 **TM** IR_2, R_1	10,5 **TM** R_1, IM	10,5 **TM** IR_1, IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1, Irr_2	18,0 **LDEI** Ir_1, Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2, Irr_1	18,0 **LDEI** Ir_2, Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1, r_2	6,5 **CP** r_1, Ir_2	10,5 **CP** R_2, R_1	10,5 **CP** IR_2, R_1	10,5 **CP** R_1, IM	10,5 **CP** IR_1, IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1, r_2	6,5 **XOR** r_1, Ir_2	10,5 **XOR** R_2, R_1	10,5 **XOR** IR_2, R_1	10,5 **XOR** R_1, IM	10,5 **XOR** IR_1, IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1, Irr_2	18,0 **LDCI** Ir_1, Irr_2				10,5 **LD** r_1, x, R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2, Irr_1	18,0 **LDCI** Ir_2, Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2, x, R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1, Ir_2	10,5 **LD** R_2, R_1	10,5 **LD** IR_2, R_1	10,5 **LD** R_1, IM	10,5 **LD** IR_1, IM								6,5 **CCF**
F	8,5 **SWAP** R_1	8,5 **SWAP** IR_1		6,5 **LD** Ir_1, r_2		10,5 **LD** R_2, IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R_1 or r_1 = Dst Address
R_2 or r_2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- □ $+4.75\ V \leq V_{CC} \leq +5.25\ V$
- □ GND = 0 V
- □ $0°C \leq T_A \leq +70°C$*

*See Ordering Information section for package temperature range and product number.

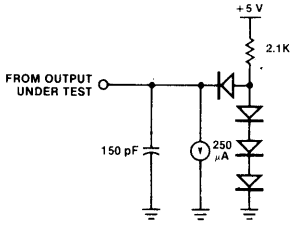


Figure 15. Test Load 1

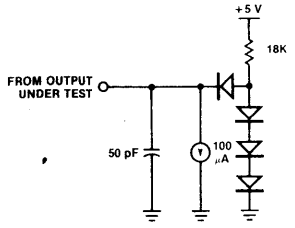


Figure 16. Test Load 2

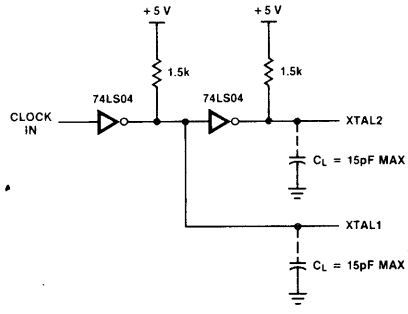


Figure 17. TTL External Clock Interface Circuit
(Both the clock and its complement are required)

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator	
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator	
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V		
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V		
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA	1
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA	1
I_{IL}	Input Leakage	-10	10	µA	$0\ V \leq V_{IN} \leq +5.25\ V$	
I_{OL}	Output Leakage	-10	10	µA	$0\ V \leq V_{IN} \leq +5.25\ V$	
I_{IR}	Reset Input Current		-50	µA	V_{CC} = +5.25 V, V_{RL} = 0 V	
I_{CC}	V_{CC} Supply Current		180	mA		
I_{MM}	V_{MM} Supply Current		10	mA	Power Down Mode	
V_{MM}	Backup Supply Voltage	3	V_{CC}	V	Power Down	

1. For A_0-A_{11}, $\overline{MDS}$, $\overline{SYNC}$, SCLK and IACK on the Z8612 version, I_{OH} = -100 µA and I_{OL} = 1.0 mA.

External I/O or Memory Read and Write Timing

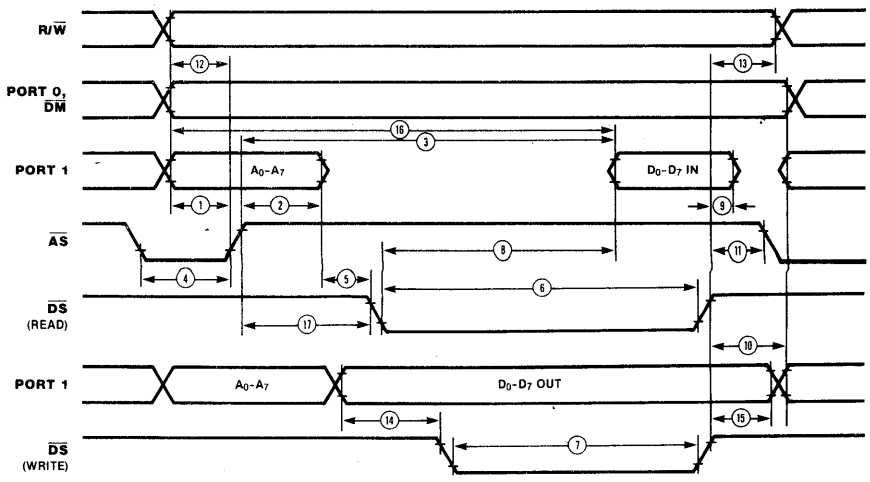


Figure 18. External I/O or Memory Read/Write

No.	Symbol	Parameter	Z8611/2/3 Min	Z8611/2/3 Max	Z8611/2/3-12 Min	Z8611/2/3-12 Max	Notes*†
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		55		1,2,3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/$\overline{W}$ Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		1,2,3

NOTES:
1. Test Load 1
2. Timing numbers given are for minimum TpC.
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 TpC.
5. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* All units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Additional Timing Table

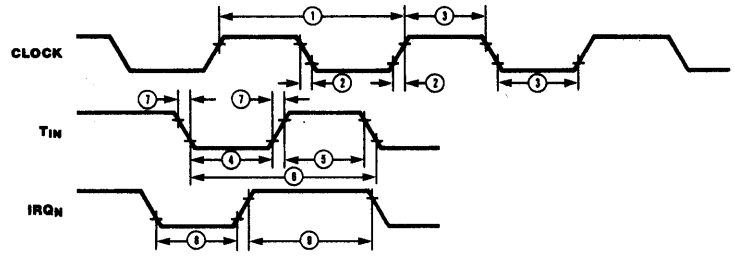


Figure 19. Additional Timing

No.	Symbol	Parameter	Z8611/2/3 Min	Z8611/2/3 Max	Z8611/2/3-12 Min	Z8611/2/3-12 Max	Notes*†
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin,TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIL	Interrupt Request Input Low Time	3TpC		3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:
1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0"
3. Interrupt request via Port 3 ($P3_1$-$P3_3$).
4. Interrupt request via Port 3 ($P3_0$).
* Units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Z8612, Z8613 Memory Port Timing

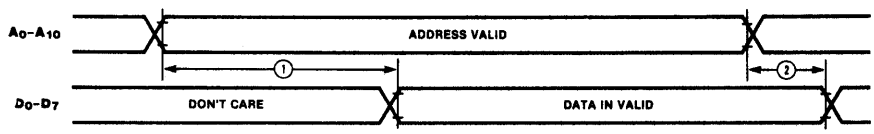


Figure 20. Memory Port Timing

No.	Symbol	Parameter	Z8611/2/3 Min	Z8611/2/3 Max	Z8611/2/3-12 Min	Z8611/2/3-12 Max	Notes*
1	TdA(DI)	Address Valid to Data Input Delay		460		320	1,2
2	ThDI(A)	Data In Hold Time	0		0		1

NOTES:
1. Test Load 2
2. This is a Clock-Cycle-Dependent parameter. For clock frequencies other than the maximum, use the following formula:
Z8611/2/3 = 5 TpC − 165
Z8611/2/3-12 = 5 TpC − 95

* Units are nanoseconds unless otherwise specified; timings are preliminary and subject to change.

Handshake Timing

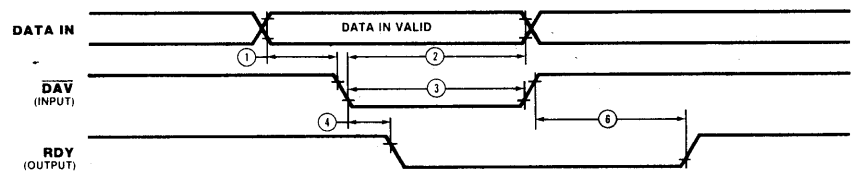


Figure 21a. Input Handshake

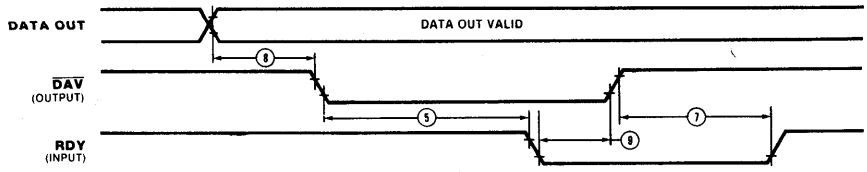


Figure 21b. Output Handshake

No.	Symbol	Parameter	Z8611/2/3 Min	Z8611/2/3 Max	Z8611/2/3-12 Min	Z8611/2/3-12 Max	Notes*†
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAV0rRDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake
4. All timing regerences use 2.0 V for a logic "1" and 0.8 V for a logic "0"

* Units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8611/2/3 Equation	Z8611/2/3-12 Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

* Add 2TpC when using extended memory timing

Ordering Information

Product Number	Package/ Temp	Speed	Description
Z8611	CE	8.0 MHz	Z8 MCU (4K ROM, 40-pin)
Z8611	CS	8.0 MHz	Same as above
Z8611	PE	8.0 MHz	Same as above
Z8611	PS	8.0 MHz	Same as above
Z8611-12	CE	12.0 MHz	Z8 MCU (4K ROM, 40-pin)
Z8611-12	CS	12.0 MHz	Same as above
Z8611-12	PE	12.0 MHz	Same as above
Z8611-12	PS	12.0 MHz	Same as above
Z8612	CE	8.0 MHz	Z8 MCU (4K XROM, 64-pin)
Z8612	CS	8.0 MHz	Same as above
Z8612	PE	8.0 MHz	Z8 MCU (4K XROM, 64-pin)
Z8612-12	PS	8.0 MHz	Same as above
	CE	12.0 MHz	Same as above
Z8612-12	CS	12.0 MHz	Same as above
Z8612-12	PE	12.0 MHz	Same as above
Z8612-12	PS	12.0 MHz	Same as above
Z8613	RS	8.0 MHz	Z8 MCU (4K XROM, Prototyping Device, 40-pin)
Z8613-12	RS	12.0 MHz	Same as above

NOTES: C = Ceramic, P = Plastic, R = Prototyping Device; E = −40° to +85°C, S = 0°C to +70°C.

Z8® Family
Z8671 MCU with BASIC/Debug Interpreter

Zilog

Product Specification

September 1983

Features

- The Z8671 MCU is a complete microcomputer preprogrammed with a BASIC/Debug interpreter. Interaction between the interpreter and its user is provided through an on-board UART.
- BASIC/Debug can directly address the Z8671's internal registers and all external memory. It provides quick examination and modification of any external memory location or I/O port.
- The BASIC/Debug interpreter can call machine language subroutines to increase execution speed.
- The Z8671's auto start-up capability allows a program to be executed on power-up or Reset without operator intervention.
- Single +5 V power supply—all I/O pins TTL-compatible.
- 8 MHz/12 MHz

General Description

The Z8671 Single-Chip Microcomputer (MCU) is one of a line of preprogrammed chips—in this case with a BASIC/Debug interpreter in ROM—offered by Zilog. As a member of the Z8 Family of microcomputers, it offers the same abundance of resources as the other Z8 microcomputers.

Because the BASIC/Debug interpreter is already part of the chip circuit, programming is made much easier. The Z8671 MCU thus offers a combination of software and hardware that is ideal for many industrial control applications. The Z8671 MCU allows fast hardware tests and bit-by-bit examination and modification of memory location, I/O ports, or registers. It also allows bit manipulation and logical operations. A self-contained line editor supports interactive debugging, further speeding up program development.

The BASIC/Debug interpreter, a subset of Dartmouth BASIC, operates with three kinds of memory: on-chip registers and external ROM or RAM. The BASIC/Debug interpreter is located in the 2K bytes of on-chip ROM.

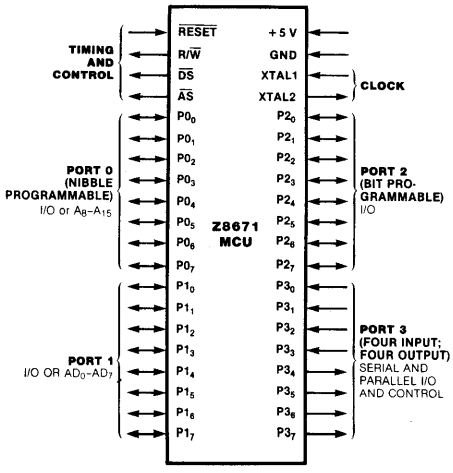


Figure 1. Pin Functions

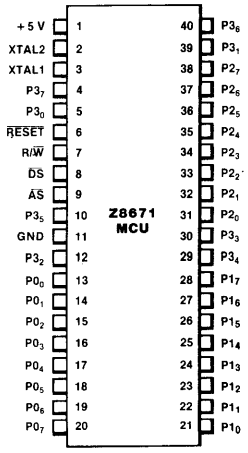


Figure 2. Pin Assignments

General Description (Continued)

Additional features of the Z8671 MCU include the ability to call machine language subroutines to increase execution speed and the ability to have a program execute on power-up or Reset, without operator intervention.

Maximum memory addressing capabilities include 62K bytes of external program memory and 62K bytes of data memory with program storage beginning at location 800 hex. This provides up to 124K bytes of useable memory space. Very few 8-bit microcomputers can directly access this amount of memory.

Each Z8671 Microcomputer has 32 I/O lines, a 144-byte register file, an on-board UART, and two counter/timers.

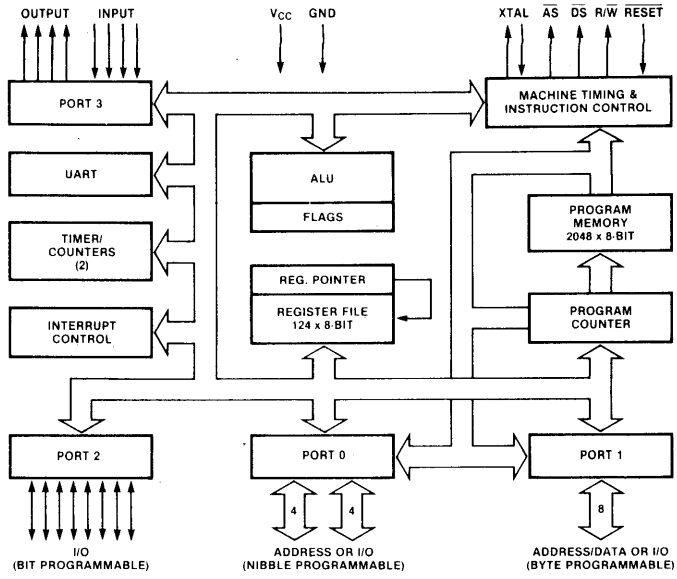


Figure 3. Functional Block Diagram

Architecture

Z8671 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure, and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8671 fulfills this with 32 pins dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an address/data bus for interfacing external memory.

Because the multiplexed address/data bus is merged with the I/O-oriented ports, the Z8671 can assume many different memory and I/O configurations. These configurations range from a self-contained microcomputer to a microprocessor that can address 124K bytes of external memory.

Three basic address spaces are available to support this wide range of configurations: program memory (internal and external), data memory (external) and the register file (internal). The 144-byte random-access register file is composed of 124 general-purpose registers, four I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate.

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$. Under program control, $\overline{AS}$ can be placed in the high-impedance state along with Ports 0 and 1, Data Strobe and Read/Write.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 32 lines are divided into four 8-bit I/O ports that can be configured under program control for I/O or external memory interface.

$\overline{RESET}$. *Reset* (input, active Low). $\overline{RESET}$ initializes the Z8671. When $\overline{RESET}$ is deactivated, program execution begins from internal program location $000C_H$.

R/$\overline{W}$. *Read/Write* (output). R/$\overline{W}$ is Low when the Z8671 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal (8 or 12 MHz maximum) or an external single-phase clock (8 or 12 MHz maximum) to the on-chip clock oscillator and buffer.

Address Spaces

Program Memory. The Z8671's 16-bit program counter can address 64K bytes of program memory space. Program memory consists of 2K bytes of internal ROM and up to 62K bytes of external ROM, EPROM, or RAM. The first 12 bytes of program memory are reserved for interrupt vectors (Figure 4). These locations contain six 16-bit vectors that correspond to the six available interrupts. The BASIC/Debug interpreter is located in the 2K bytes of internal ROM. The interpreter begins at address 12 and extends to 2047.

Data Memory. The Z8671 can address up to 62K bytes of external data memory beginning at location 2048 (Figure 5). External data memory may be included with, or separated from, the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish data and program memory space.

Register File. The 144-byte register file may be accessed by BASIC programs as memory locations 0-127 and 240-255. The register file includes four I/O port registers (R0-R3), 124 general-purpose registers (R4-R127), and 16 control and status registers (Figure 6).

The BASIC/Debug Interpreter uses many of the general-purpose registers as pointers, scratch workspace, and internal variables. Consequently, these registers cannot be used by a machine language subroutine or other user programs. On power-up/Reset, BASIC/Debug searches for external RAM memory and

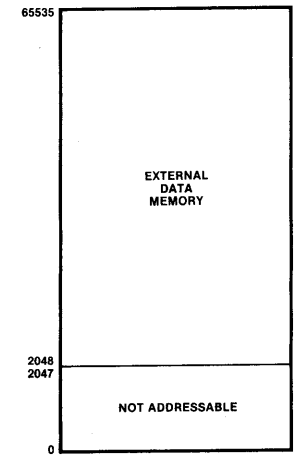


Figure 5. Data Memory Map

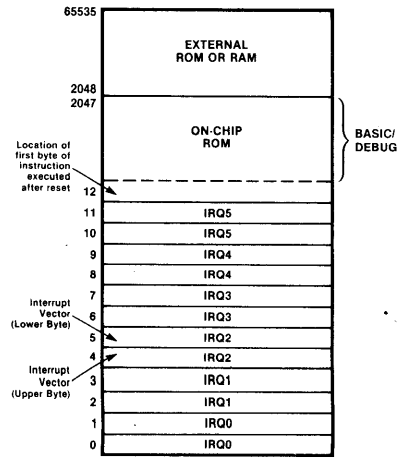


Figure 4. Program Memory Map

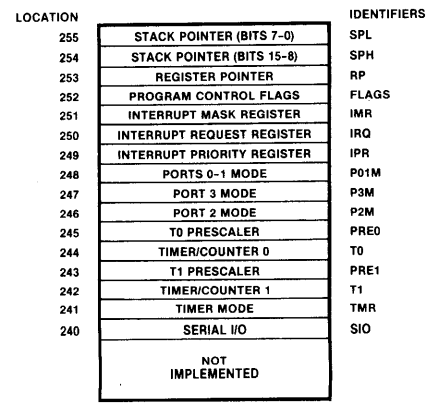


Figure 6. Control and Status Registers

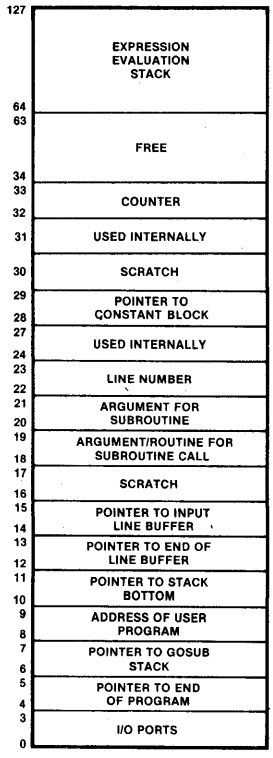


Figure 7a. General-Purpose Registers with External RAM

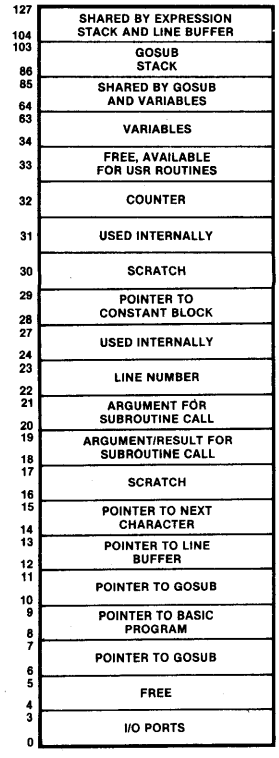


Figure 7b. General-Purpose Registers without External RAM

checks for an auto start-up program. In a non-destructive method, memory is tested at relative location xxFD(hex). When BASIC/Debug discovers RAM in the system, it initializes the pointer registers to mark the boundaries between areas of memory that are assigned specific uses. The top page of RAM is allocated for the line buffer, variable storage, and the GOSUB stack. Figure 7a illustrates the contents of the general-purpose registers in the Z8671 system with external RAM. When BASIC/Debug tests memory and finds no RAM, it uses an internal stack and shares register space with the input line buffer and variables. Figure 7b illustrates the contents of the general-purpose registers in the Z8671 system without external RAM.

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory between location 2048 and 65535. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

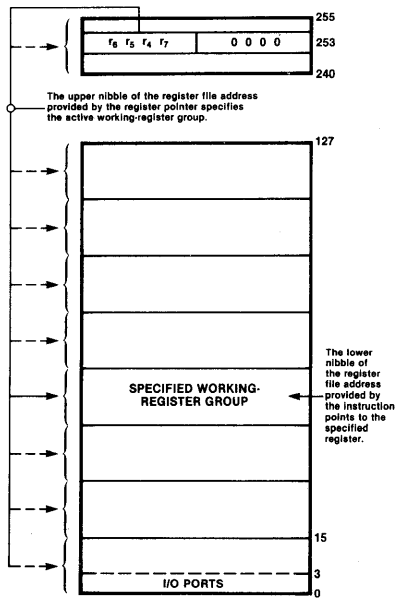


Figure 8. The Register Pointer

Address Spaces (Continued)

Register Addressing. Z8671 instructions can access registers directly or indirectly with an 8-bit address field. The Z8671 also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each group consisting of 16 contiguous registers (Figure 8). The Register Pointer addresses the starting location of the active working-register group.

Program Execution

Automatic Start-up. The Z8671 has an automatic start-up capability which allows a program stored in ROM to be executed without operator intervention. Automatic execution occurs on power-on or Reset when the program is stored at address 1020 (hex).

Execution Modes. The Z8671's BASIC/Debug Interpreter operates in two execution modes: Run and Immediate. Programs are edited and interactively debugged in the Immediate mode. Some BASIC/Debug commands are used almost exclusively in this mode. The Run mode is entered from the Immediate mode by entering the command RUN. If there is a program in RAM, it is executed. The system returns to the Immediate mode when program execution is complete or interrupted by an error.

Interactive Debugging

Interactive debugging is accomplished with the self-contained line editor which operates in the Immediate mode. In addition to changing program lines, the editor can correct an immediate command before it is executed. It also allows the correction of typing and other errors as a program is entered.

BASIC/Debug allows interruptions and changes during a program run to correct errors and add new instructions without disturbing the sequential execution of the program. A program run is interrupted with the use of the escape key. The run is restarted with a GOTO command (followed by the appropriate line number) after the desired changes are entered. The same procedure is used to enter corrections after BASIC/Debug returns an error.

Commands

BASIC/Debug recognizes 15 command keywords. For detailed instructions of command usage, refer to the *BASIC/Debug Software Reference Manual* (#03-3149-02).

GO	The GO command unconditionally branches to a machine language subroutine. This statement is similar to the USR function except that no value is returned by the assembly language routine.
GOSUB	GOSUB unconditionally branches to a subroutine at a line number specified by the user.
GOTO	GOTO unconditionally changes the sequence of program execution (branches to a line number).
IF/THEN	This command is used for conditional operations and branches.
INPUT/IN	These commands request information from the user with the prompt "?", then read the input values (which must be separated by commas) from the keyboard, and store them in the indicated variables. INPUT discards any values remaining in the buffer from previous IN, INPUT, or RUN statements, and requests new data from the operator. IN uses any values left in the buffer first, then requests new data.
LET	LET assigns the value of an expression to a variable or memory location.
LIST	This command is used in the interactive mode to generate a listing of program lines stored in memory on the terminal device.
NEW	The NEW command resets pointer R10-11 to the beginning of user memory, thereby marking the space as empty and ready to store a new program.
PRINT	PRINT lists its arguments, which may be text messages or numerical values, on the output terminal.
REM	This command is used to insert explanatory messages into the program.
RETURN	This command returns control to the line following a GOSUB statement.
RUN	RUN initiates sequential execution of all instructions in the current program.
STOP	STOP ends program execution and clears the GOSUB stack.

Functions

BASIC/Debug supports two functions: AND and USR.

The AND function performs a logical AND. It can be used to mask, turn off, or isolate bits. This function is used in the following format:

AND (expression, expression)

The two expressions are evaluated, and their bit patterns are ANDed together. If only one value is included in the parentheses, it is ANDed with itself. A logical OR can also be performed by complementing the AND function. This is accomplished by subtracting each expression from -1. For example, the function below is equivalent to the OR of A and B.

-1-AND(-1-A, -1-B)

The USR function calls a machine language subroutine and returns a value. This is useful for applications in which a subroutine can be performed more quickly and efficiently in machine language than in BASIC/Debug.

The address of the first instruction of the subroutine is the first argument of the USR function. The address can be followed by one or two values to be processed by the subroutine. In the following example, BASIC/Debug executes the subroutine located at address 2000 using values literal 256 and variable C.

USR(%2000,256,C)

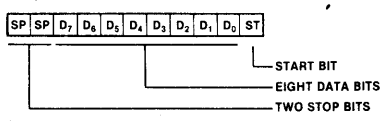
The resulting value is stored in Registers 18-19.

Serial Input/ Output

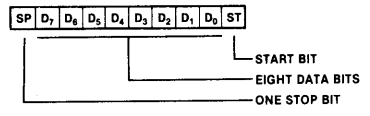
Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second for 8MHz, and a maximum rate of 94.8K bits/second for 12MHz parts.

The Z8671 automatically adds a start bit and two stop bits to transmitted data (Figure 9). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

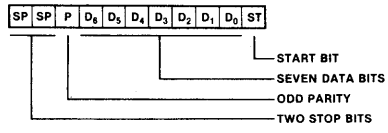
Received data must have a start bit, eight data bits, and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.



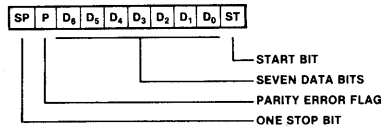
Transmitted Data
(No Parity)



Received Data
(No Parity)



Transmitted Data
(With Parity)



Received Data
(With Parity)

Figure 9. Serial Data Formats

I/O Ports

The Z8671 has 32 lines dedicated to input and output. These lines are grouped into four ports of eight lines each and are configurable as input, output or address/data. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 can be programmed as a byte I/O port or as an address/data port for interfacing external memory. When used as an I/O port, Port 1 may be placed under handshake control. In this configuration, Port 3 lines $P3_3$ and $P3_4$ are used as the handshake controls RDY_1 and $\overline{DAV}_1$ (Ready and Data Available).

Memory locations greater than 2048 are referenced through Port 1. To interface external memory, Port 1 must be programmed for the multiplexed Address/Data mode. If more than 256 external locations are required, Port 0 must output the additional lines.

Port 1 can be placed in the high-impedance state along with Port 0, $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$, allowing the Z8671 to share common resources in multiprocessor and DMA applications. Data transfers can be controlled by assigning $P3_3$ as a Bus Acknowledge input and $P3_4$ as a Bus Request output.

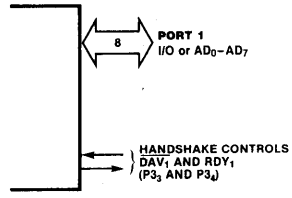


Figure 10a. Port 1

Port 0 can be programmed as a nibble I/O port, or as an address port for interfacing external memory. When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls $\overline{DAV}_0$ and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibble) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing. When Port 0 nibbles are defined as address bits, they can be set to the high-impedance state along with Port 1 and the control signals $\overline{AS}$, $\overline{DS}$ and R/$\overline{W}$.

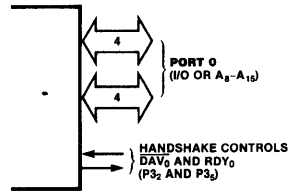


Figure 10b. Port 0

Port 2 bits can be programmed independently as input or output. The port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Ports 0 and 1, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines $\overline{DAV}_2$ and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

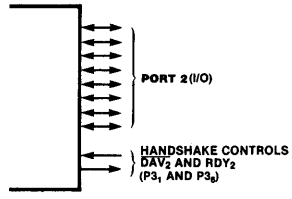


Figure 10c. Port 2

Port 3 lines can be configured as I/O or control lines. In either case, the direction of the eight lines is fixed as four input ($P3_0$-$P3_3$) and four output ($P3_4$-$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out respectively.

Port 3 can also provide the following control functions: handshake for Ports 0, 1 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0-IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

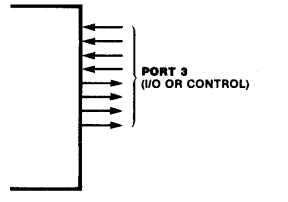


Figure 10d. Port 3

Counter/ Timers

The Z8671 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable; it can be either the internal microprocessor clock (4 MHz maximum for the 8 MHz device and 6 MHz maximum for the 12 MHz device) divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or non-retriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

Interrupts

The Z8671 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$-$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8671 interrupts are vectored; however, the internal UART operates in a polling fashion. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

The BASIC/Debug Interpreter does not process interrupts. Interrupts are vectored through locations in internal ROM which point to addresses 1000-1011 (hex). To process interrupts, jump instructions can be entered to the interrupt handling routines at the appropriate addresses as shown in Table 1.

Address (hex)	Contains Jump Instruction and Subroutine Address for:
1000-1002	IRQ0
1003-1005	IRQ1
1006-1008	IRQ2
1009-100B	IRQ3
100C-100E	IRQ4
100F-1011	IRQ5

Table 1. Interrupt Jump Instructions

Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitance (C_L = 15 pF maximum) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel resonant
- Fundamental type, 8/12 MHz maximum
- Series resistance, $R_s \leq 100\ \Omega$
- 8 MHz maximum for Z8671
- 12 MHz maximum for Z8671-12

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats

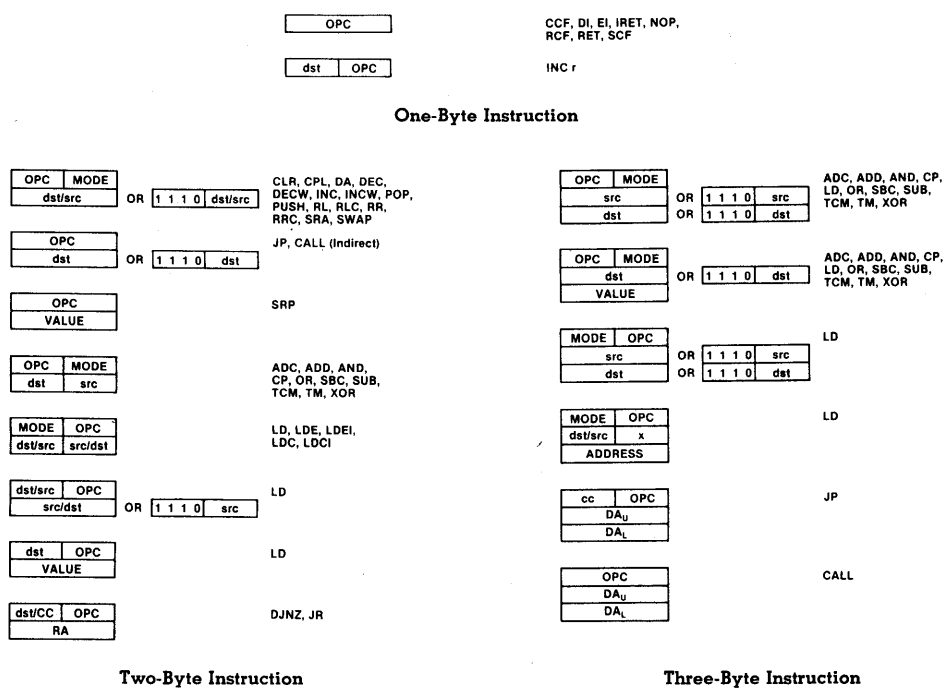
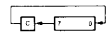
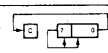
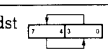


Figure 11. Instruction Formats

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR(7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R R IR IR	Im R r X r Ir r R IR Im Im R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst 	R IR		90 91	* * * * - -
RLC dst 	R IR		10 11	* * * * - -
RR dst 	R IR		E0 E1	* * * * - -
RRC dst 	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst 	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst 	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

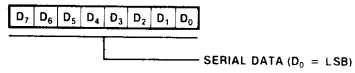
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

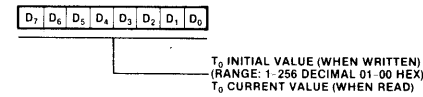
For example, the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

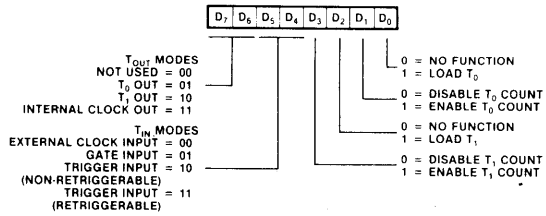
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



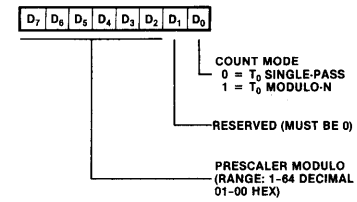
R244 T0
Counter/Timer 0 Register
($F4_H$; Read/Write)



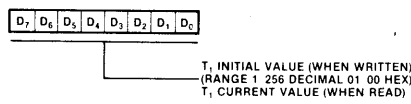
R241 TMR
Timer Mode Register
($F1_H$; Read/Write)



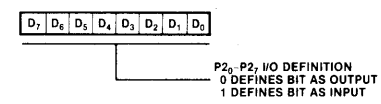
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



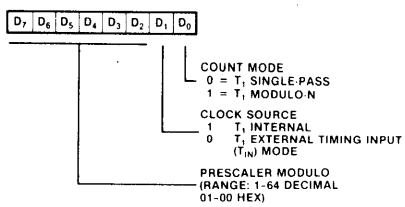
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

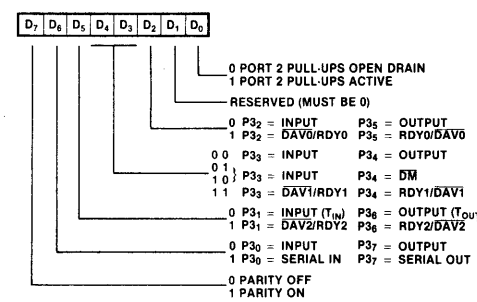
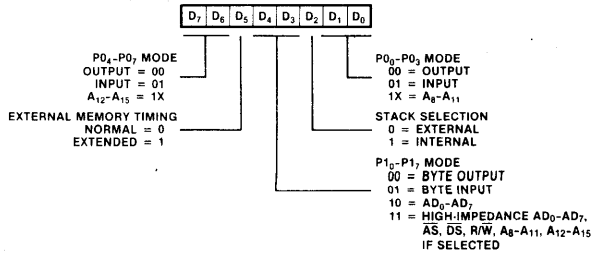
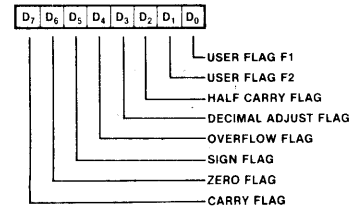


Figure 12. Control Registers

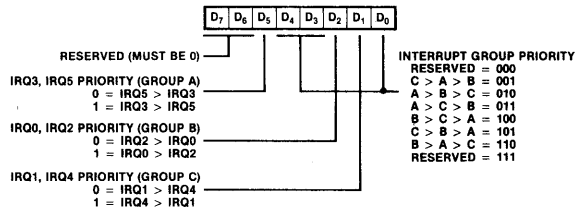
R248 P01M
Port 0 and 1 Mode Register
($F8_H$; Write Only)



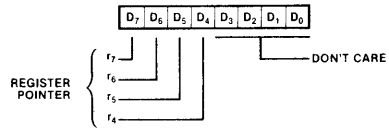
R252 FLAGS
Flag Register
(FC_H; Read/Write)



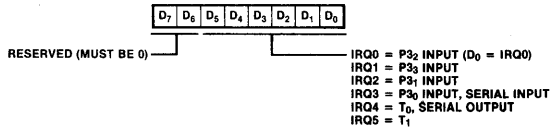
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



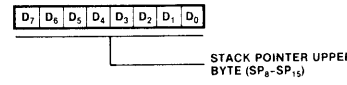
R253 RP
Register Pointer
(FD_H; Read/Write)



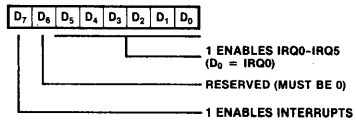
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

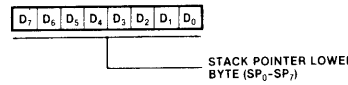
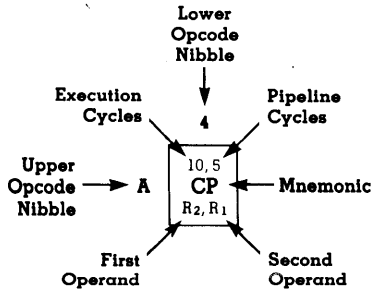


Figure 12. Control Registers (Continued)

Z8671 Opcode Map

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1, r_2	6,5 **ADD** r_1, Ir_2	10,5 **ADD** R_2, R_1	10,5 **ADD** IR_2, R_1	10,5 **ADD** R_1, IM	10,5 **ADD** IR_1, IM	6,5 **LD** r_1, R_2	6,5 **LD** r_2, R_1	12/10,5 **DJNZ** r_1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r_1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r_1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1, r_2	6,5 **ADC** r_1, Ir_2	10,5 **ADC** R_2, R_1	10,5 **ADC** IR_2, R_1	10,5 **ADC** R_1, IM	10,5 **ADC** IR_1, IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1, r_2	6,5 **SUB** r_1, Ir_2	10,5 **SUB** R_2, R_1	10,5 **SUB** IR_2, R_1	10,5 **SUB** R_1, IM	10,5 **SUB** IR_1, IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1, r_2	6,5 **SBC** r_1, Ir_2	10,5 **SBC** R_2, R_1	10,5 **SBC** IR_2, R_1	10,5 **SBC** R_1, IM	10,5 **SBC** IR_1, IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1, r_2	6,5 **OR** r_1, Ir_2	10,5 **OR** R_2, R_1	10,5 **OR** IR_2, R_1	10,5 **OR** R_1, IM	10,5 **OR** IR_1, IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1, r_2	6,5 **AND** r_1, Ir_2	10,5 **AND** R_2, R_1	10,5 **AND** IR_2, R_1	10,5 **AND** R_1, IM	10,5 **AND** IR_1, IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1, r_2	6,5 **TCM** r_1, Ir_2	10,5 **TCM** R_2, R_1	10,5 **TCM** IR_2, R_1	10,5 **TCM** R_1, IM	10,5 **TCM** IR_1, IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1, r_2	6,5 **TM** r_1, Ir_2	10,5 **TM** R_2, R_1	10,5 **TM** IR_2, R_1	10,5 **TM** R_1, IM	10,5 **TM** IR_1, IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1, Irr_2	18,0 **LDEI** Ir_1, Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2, Irr_1	18,0 **LDEI** Ir_2, Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1, r_2	6,5 **CP** r_1, Ir_2	10,5 **CP** R_2, R_1	10,5 **CP** IR_2, R_1	10,5 **CP** R_1, IM	10,5 **CP** IR_1, IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1, r_2	6,5 **XOR** r_1, Ir_2	10,5 **XOR** R_2, R_1	10,5 **XOR** IR_2, R_1	10,5 **XOR** R_1, IM	10,5 **XOR** IR_1, IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1, Irr_2	18,0 **LDCI** Ir_1, Irr_2				10,5 **LD** r_1, x, R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2, Irr_1	18,0 **LDCI** Ir_2, Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2, x, R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1, IR_2	10,5 **LD** R_2, R_1	10,5 **LD** IR_2, R_1	10,5 **LD** R_1, IM	10,5 **LD** IR_1, IM								6,5 **CCF**
F	8,5 **SWAP** R_1	8,5 **SWAP** IR_1		6,5 **LD** Ir_1, r_2		10,5 **LD** R_2, IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R_1 or r_1 = Dst Address
R_2 or r_2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T_A ≤ +70°C

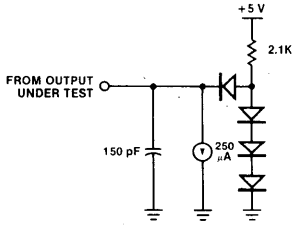


Figure 13. Test Load 1

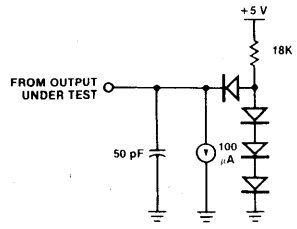


Figure 14. Test Load 2

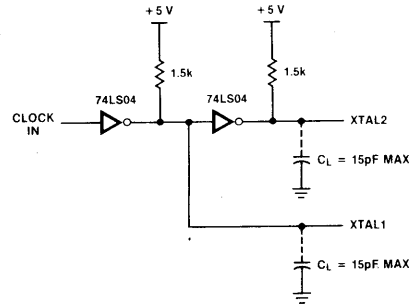


Figure 15. TTL External Clock Interface Circuit
(Both the clock and its complement are required)

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage	-10	10	µA	0 V ≤ V_{IN} ≤ +5.25 V
I_{OL}	Output Leakage	-10	10	µA	0 V ≤ V_{IN} ≤ +5.25 V
I_{IR}	Reset Input Current		-50	µA	V_{CC} = +5.25 V, V_{RL} = 0 V
I_{CC}	V_{CC} Supply Current		180	mA	
I_{MM}	V_{MM} Supply Current		10	mA	Power Down Mode
V_{MM}	Backup Supply Voltage	3	V_{CC}	V	Power Down

External I/O or Memory Read/Write

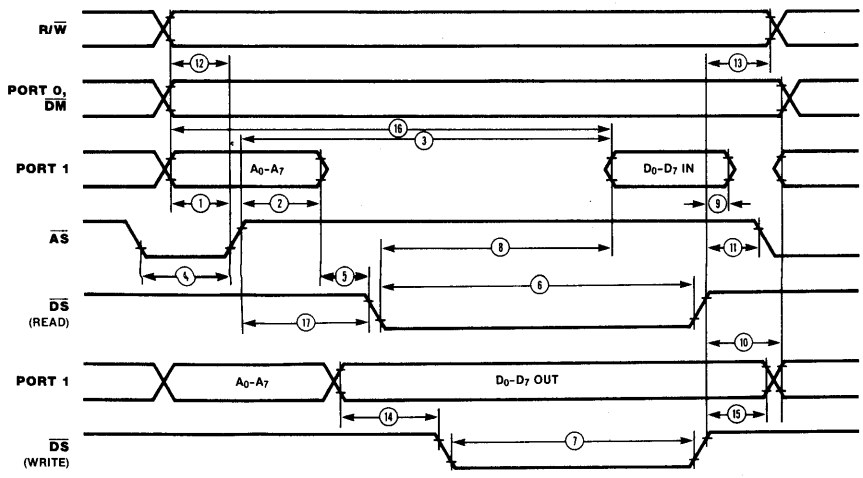


Figure 15. External I/O or Memory Read/Write

No.	Symbol	Parameter	Z8671 Min	Z8671 Max	Z8671-12 Min	Z8671-12 Max	Notes*†
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		55		1,2,3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/$\overline{W}$ Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		1,2,3

NOTES:
1. Test Load 1
2. Timing numbers given are for minimum TpC.
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 TpC.
5. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* All units in nanoseconds (ns).
† All timings are preliminary and subject to change.

Additional Timing

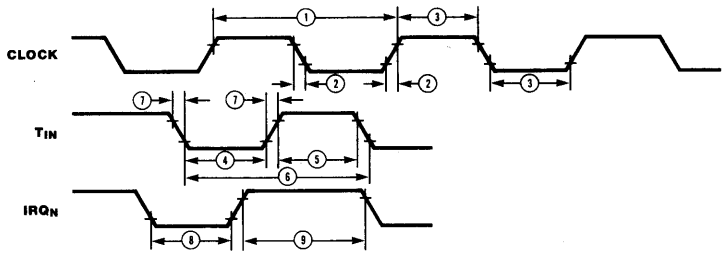


Figure 16. Additional Timing

No.	Symbol	Parameter	Z8671 Min	Z8671 Max	Z8671-12 Min	Z8671-12 Max	Notes*†
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	8TpC		8TpC		2
7	TrTin,TfTin	Timer Input Rise And Fall Times		100		100	2
8a	TwIL	Interrupt Request Input Low Time	100		70		2,3
8b	TwIL	Interrupt Request Input Low Time	3TpC		3TpC		2,4
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:
1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".
3. Interrupt request via Port 3 ($P3_1$-$P3_3$).
4. Interrupt request via Port 3 ($P3_0$).
* Units in nanoseconds (ns).
† All timings are preliminary and subject to change.

Memory Port Timing

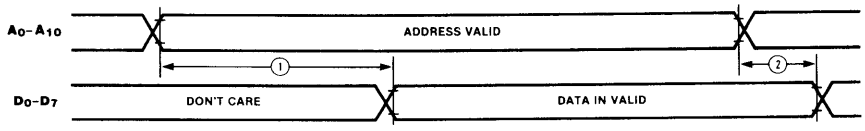


Figure 17. Memory Port Timing

No.	Symbol	Parameter	Z8671 Min	Z8671 Max	Z8671-12 Min	Z8671-12 Max	Notes*†
1	TdA(DI)	Address Valid to Data Input Delay		460		320	1,2
2	ThDI(A)	Data In Hold Time	0		0		1

NOTES:
1. Test Load 2
2. This is a Clock-Cycle-Dependent parameter. For clock frequencies other than the maximum, use the following formula:
Z8671 = 5 TpC − 165
Z8671-12 = 5 TpC − 95

* Units are nanoseconds unless otherwise specified; timings are preliminary and subject to change.

Handshake Timing

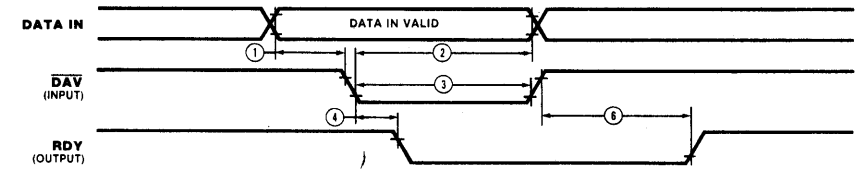


Figure 17a. Input Handshake

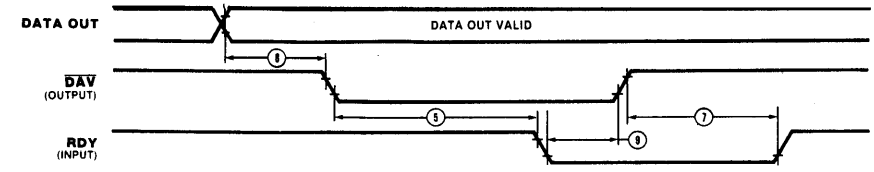


Figure 17b. Output Handshake

No.	Symbol	Parameter	Z8671 Min	Z8671 Max	Z8671-12 Min	Z8671-12 Max	Notes*†
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAV0rRDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake
4. All timing regerences use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* Units in nanoseconds (ns).
† All timings are preliminary and subject to change.

Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8671 Equation	Z8671-12 Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

* Add 2TpC when using extended memory timing

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8671	PE	8.0 MHz	Z8 MCU with BASIC/Debug Interpreter (40-pin)	Z8671-12	PE	12.0 MHZ	Z8 MCU with BASIC/Debug Interpreter (40-pin)
Z8671	PS	8.0 MHz	Same as above	Z8671-12	PS	12.0 MHz	Same as above
Z8671	CE	8.0 MHz	Same as above	Z8671-12	CE	12.0 MHz	Same as above
Z8671	CS	8.0 MHz	Same as above	Z8671-12	CS	12.0 MHz	Same as above

NOTES: C = Ceramic; P = Plastic; E = -40°C to +85°C, S = 0°C to +70°C.

Z8® Z8681/82 ROMless Microcomputer

Zilog

Product Specification

September 1983

Features

- Complete microcomputer, 24 I/O lines, and up to 64K bytes of addressable external space each for program and data memory.
- 143-byte register file, including 124 general-purpose registers, three I/O port registers, and 16 status and control registers.
- Vectored, priority interrupts for I/O, counter/timers, and UART.
- On-chip oscillator that accepts crystal or external clock drive.
- Full-duplex UART and two programmable 8-bit counter/timers, each with a 6-bit programmable prescaler.
- Register Pointer so that short, fast instructions can access any one of the nine working-register groups.
- Low-power standby option that retains contents of general-purpose registers.
- Single +5 V power supply—all I/O pins TTL compatible.
- Available in 8 and 12 MHz versions.

General Description

The Z8681and Z8682 are ROMless versions of the Z8 single-chip microcomputer. The Z8682 is usually more cost effective. These products differ only slightly and can be used interchangeably with proper system design to provide maximum flexibility in meeting price and delivery needs. The Z8681/82 offers all the outstanding features of the Z8 family architecture except an on-chip program ROM. Use of external memory rather than a preprogrammed ROM enables this Z8 microcomputer to be used in low volume applications or where code flexibility is required.

The Z8681/82 can provide up to 16 output address lines, thus permitting an address space of up to 64K bytes of data or program memory. Eight address outputs (AD_0-AD_7) are provided by a multiplexed, 8-bit, Address/Data bus. The remaining 8 bits can be provided by the software configuration of Port 0 to output address bits A_8-A_{15}.

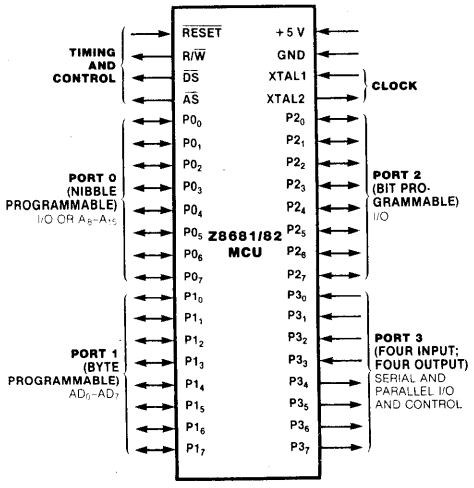


Figure 1. Pin Functions

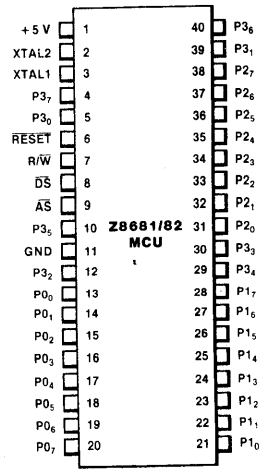


Figure 2. Pin Assignments

General Description (Continued)

Available address space can be doubled (up to 128K bytes for the Z8681 and 124K bytes for the Z8682) by programming bit 4 of Port 3 ($P3_4$) to act as a data memory select output (DM). The two states of DM together with the 16 address outputs can define separate data and memory address spaces of up to 64K/62Kbytes each.

There are 143 bytes of RAM located on-chip and organized as a register file of 124 general-purpose registers, 16 control and status registers, and three I/O port registers. This register file can be divided into nine groups of 16 working registers each. Configuring the register file in this manner allows the use of short format instructions; in addition, any of the individual registers may be accessed directly.

The pin functions and the pin assignments of the Z8681/82 40-pin package are illustrated in Figures 1 and 2, respectively.

Architecture

Z8681/82 architecture is characterized by a flexible I/O scheme, an efficient register and address space structure and a number of ancillary features that are helpful in many applications.

Microcomputer applications demand powerful I/O capabilities. The Z8681/82 fulfills this with 24 pins available for input and output. These lines are grouped into three ports of eight lines each and are configurable under software control to provide timing, status signals, serial or parallel I/O with or without handshake, and an Address bus for interfacing external memory.

Three basic address spaces are available:. program memory, data memory and the register file (internal). The 143-byte random-access register file is composed of 124 general-purpose registers, three I/O port registers, and 16 control and status registers.

To unburden the program from coping with real-time problems such as serial data communication and counting/timing, an asynchronous receiver/transmitter (UART) and two counter/timers with a large number of user-selectable modes are offered on-chip. Hardware support for the UART is minimized because one of the on-chip timers supplies the bit rate. Figure 3 shows the Z8681/82 block diagram.

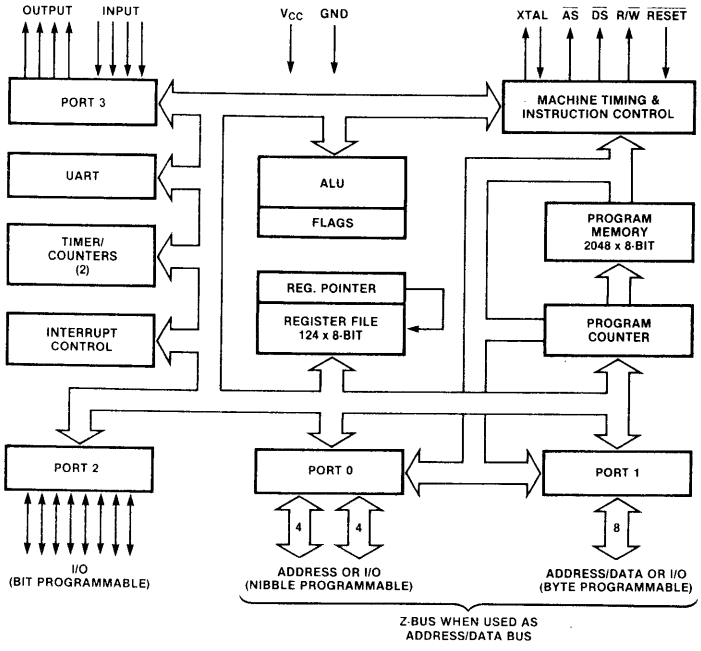


Figure 3. Functional Block Diagram

Pin Description

$\overline{AS}$. *Address Strobe* (output, active Low). Address Strobe is pulsed once at the beginning of each machine cycle. Addresses output via Port 1 for all external program or data memory transfers are valid at the trailing edge of $\overline{AS}$.

$\overline{DS}$. *Data Strobe* (output, active Low). Data Strobe is activated once for each external memory transfer.

$P0_0$-$P0_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (input/outputs, TTL-compatible). These 24 lines are divided into three 8-bit I/O ports that can be configured under program control for I/O or external memory interface (Figure 3).

$P1_0$-$P1_7$. *Address/Data Port* (bidirectional). Multiplexed address (A_0-A_7) and data (D_0-D_7) lines used to interface with program and data memory.

Pin Description (Continued)

$\overline{\text{RESET}}$.* *Reset* (input, active Low). $\overline{\text{RESET}}$ initializes the Z8681/82. When $\overline{\text{RESET}}$ is deactivated, program execution begins from program location $000C_H$ for the Z8681 and 0812_H for the Z8682.

R/$\overline{\text{W}}$. *Read/Write* (output). R/$\overline{\text{W}}$ is Low when the Z8681/82 is writing to external program or data memory.

XTAL1, XTAL2. *Crystal 1, Crystal 2* (time-base input and output). These pins connect a parallel-resonant crystal to the on-chip clock oscillator and buffer.

Summary of Z8681 and Z8682 Differences

Feature	Z8681	Z8682
Address of first instruction executed after Reset	12	2066
Addressable memory space	0-64K	2K-64K
Address of interrupt vectors	0-11	2048-2065
Reset input high voltage	TTL levels*	7.35-8.0 V
Port 0 configuration after Reset	Input, float after reset. Can be programmed as Address bits.	Output, configured as Address bits A_8-A_{15}.
External memory timing start-up configurations	Extended Timing	Normal Timing
Interrupt vectors	2 byte vectors point directly to service routines.	2 byte vectors in internal ROM point to 3 byte Jump instructions, which point to service routines.
Interrupt response time	26μsec	36μsec

*8.0 V V_{IN} max

Address Spaces

Program Memory.* The Z8681/82 addresses 64K/62K bytes of external program memory space (Figure 4).

For the Z8681, the first 12 bytes of program memory are reserved for the interrupt vectors. These locations contain six 16-bit vectors that

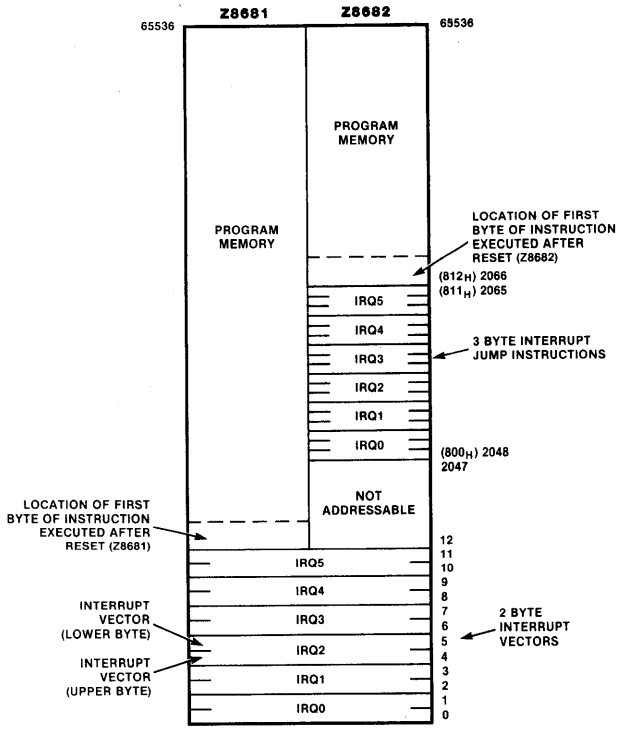


Figure 4. Z8681/82 Program Memory Map

*This feature differs in the Z8681 and Z8682

Address Spaces (Continued)

correspond to the six available interrupts. Program execution begins at location $000C_H$ after a reset.

The Z8682 has six 24-bit interrupt vectors beginning at address 0800_H. The vectors consist of Jump Absolute instructions. After a reset, program execution begins at location 0812_H for the Z8682.

Data Memory.* The Z8681/82 can address 64K/62K bytes of external data memory. External data memory may be included with or separated from the external program memory space. $\overline{DM}$, an optional I/O function that can be programmed to appear on pin $P3_4$, is used to distinguish between data and program memory space.

Register File. The 143-byte register file includes three I/O port registers (R0, R2, R3), 124 general-purpose registers (R4-R127) and 16 control and status registers (R240-R255).

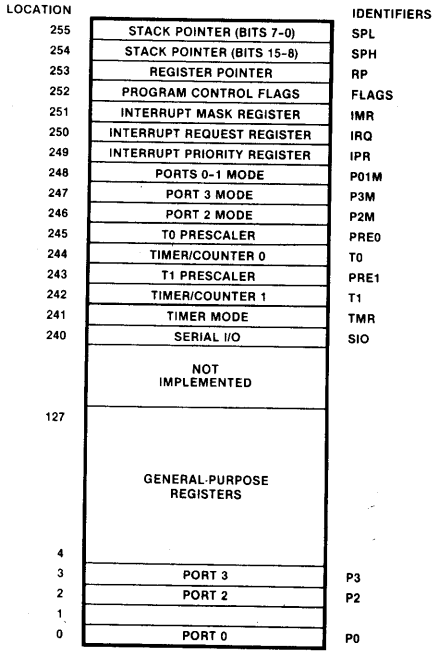


Figure 5. The Register File

These registers are assigned the address locations shown in Figure 5.

Z8681/82 instructions can access registers directly or indirectly with an 8-bit address field. This also allows short 4-bit register addressing using the Register Pointer (one of the control registers). In the 4-bit mode, the register file is divided into nine working-register groups, each occupying 16 contiguous locations (Figure 5). The Register Pointer addresses the starting location of the active working-register group (Figure 6).

Stacks. Either the internal register file or the external data memory can be used for the stack. A 16-bit Stack Pointer (R254 and R255) is used for the external stack, which can reside anywhere in data memory. An 8-bit Stack Pointer (R255) is used for the internal stack that resides within the 124 general-purpose registers (R4-R127).

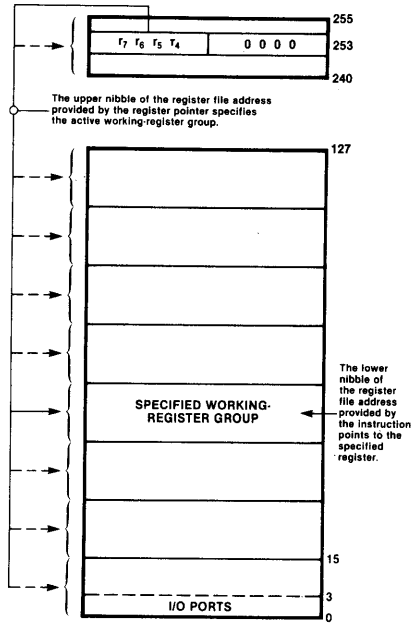


Figure 6. The Register Pointer

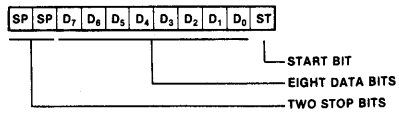
*This feature differs in the Z8681 and Z8682

Serial Input/ Output

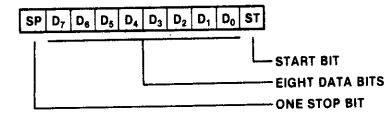
Port 3 lines $P3_0$ and $P3_7$ can be programmed as serial I/O lines for full-duplex serial asynchronous receiver/transmitter operation. The bit rate is controlled by Counter/Timer 0, with a maximum rate of 62.5K bits/second at 8 MHz and 93.75K bits/second at 12 MHz.

The Z8681/82 automatically adds a start bit and two stop bits to transmitted data (Figure 7). Odd parity is also available as an option. Eight data bits are always transmitted, regardless of parity selection. If parity is enabled, the eighth data bit is used as the odd parity bit. An interrupt request (IRQ_4) is generated on all transmitted characters.

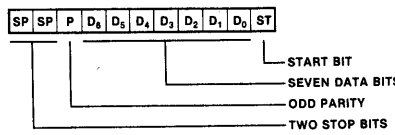
Received data must have a start bit, eight data bits and at least one stop bit. If parity is on, bit 7 of the received data is replaced by a parity error flag. Received characters generate the IRQ_3 interrupt request.



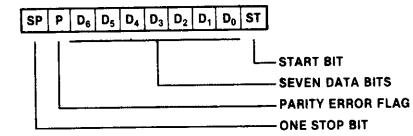
Transmitted Data
(No Parity)



Received Data
(No Parity)



Transmitted Data
(With Parity)



Received Data
(With Parity)

Figure 7. Serial Data Formats

Counter/ Timers

The Z8681/82 contains two 8-bit programmable counter/timers (T_0 and T_1), each driven by its own 6-bit programmable prescaler. The T_1 prescaler can be driven by internal or external clock sources; however, the T_0 prescaler is driven by the internal clock only.

The 6-bit prescalers can divide the input frequency of the clock source by any number from 1 to 64. Each prescaler drives its counter, which decrements the value (1 to 256) that has been loaded into the counter. When the counter reaches the end of count, a timer interrupt request—IRQ_4 (T_0) or IRQ_5 (T_1)—is generated.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. The counters can also be programmed to stop upon reaching zero (single-pass mode) or to automatically reload the initial value and continue counting (modulo-n continuous mode). The counters, but not the prescalers, can be read any time without disturbing their value or count mode.

The clock source for T_1 is user-definable; it can be either the internal microprocessor clock divided by four, or an external signal input via Port 3. The Timer Mode register configures the external timer input as an external clock, a trigger input that can be retriggerable or nonretriggerable, or as a gate input for the internal clock. The counter/timers can be programmably cascaded by connecting the T_0 output to the input of T_1. Port 3 line $P3_6$ also serves as a timer output (T_{OUT}) through which T_0, T_1 or the internal clock can be output.

I/O Ports

The Z8681/82 has 24 lines available for input and output. These lines are grouped into three ports of eight lines each and are configurable as input, output or address. Under software control, the ports can be programmed to provide address outputs, timing, status signals, serial I/O, and parallel I/O with or without handshake. All ports have active pull-ups and pull-downs compatible with TTL loads.

Port 1 is a dedicated Z-BUS compatible memory interface. The operations of Port 1 are supported by the Address Strobe ($\overline{AS}$) and Data Strobe ($\overline{DS}$) lines, and by the Read/Write ($R/\overline{W}$) and Data Memory ($\overline{DM}$) control lines. The low-order program and data memory addresses (A_0–A_7) are output through Port 1 (Figure 8) and are multiplexed with data in/out (D_0–D_7). Instruction fetch and data memory read/write operations are done through this port.

Port 1 cannot be used as a register nor can a handshake mode be used with this port.

I/O Ports (Continued)

Both the Z8681 and Z8682 wake up with the 8 bits of Port 1 configured as address outputs for external memory. If more than eight address line are required with the Z8681, additional lines can be obtained by programming Port 0 bits as address bits. The least-significant four bits of Port 0 can be configured to supply address bits A_8-A_{11} for 4K byte addressing or both nibbles of Port 0 can be configured to supply address bits A_8-A_{15} for 64K byte addressing.

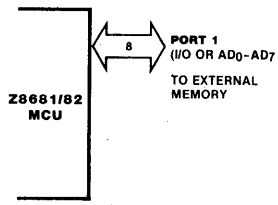


Figure 8. Port 1

Port 0* can be programmed as a nibble I/O port, or as an address port for interfacing external memory (Figure 9). When used as an I/O port, Port 0 may be placed under handshake control. In this configuration, Port 3 lines $P3_2$ and $P3_5$ are used as the handshake controls DAV_0 and RDY_0. Handshake signal assignment is dictated by the I/O direction of the upper nibble $P0_4$-$P0_7$.

For external memory references, Port 0 can provide address bits A_8-A_{11} (lower nibble) or A_8-A_{15} (lower and upper nibbles) depending on the required address space. If the address range requires 12 bits or less, the upper nibble of Port 0 can be programmed independently as I/O while the lower nibble is used for addressing.

In the Z8681,* Port 0 lines float after reset; their logic state is unknown until the execution of an initialization routine that configures Port 0.

Such an initialization routine must reside within the first 256 bytes of executable code and must be physically mapped into memory by forcing the Port 0 address lines to a known state. See Figure 10. The proper Port initialization sequence is:

1. Write initial address (A_8-A_{15}) of initialization routine to Port 0 address lines.
2. Configure Port 0 Mode Register to output A_8-A_{15} (or A_8-A_{11}).

To permit the use of slow memory, an automatic wait mode of two oscillator clock cycles is configured for the bus timing of the Z8681 after each reset. The initialization routine could include reconfiguration to eliminate this extended timing mode.

The following example illustrates the manner in which an initialization routine can be mapped in a Z8681 system with 4K of memory.

Example. In Figure 10, the initialization routine is mapped to the first 256 bytes of program memory. Pull-down resistors maintain the address lines at a logic 0 level when these lines are floating. The leakage current caused by fanout must be taken into consideration when selecting the value of the pulldown resistors. The resistor value must be large enough to allow the Port 0 output driver to pull the line to a logic one. Generally, pulldown resistors are incompatible with TTL loads. If Port 0 drives into TTL input loads (I_{LOW} = 1.6 ma) the external resistors should be tied to V_{CC} and the initialization routine put in address space $FF00_H$-$FFFF_H$.

In the Z8682,* Port 0 lines are configured as address lines A_8-A_{15} after a Reset. If one or both nibbles are needed for I/O operation, they must be configured by writing to the Port 0 Mode register. The Z8682 is in the fast memory timing mode after Reset, so the initialization routine must be in fast memory.

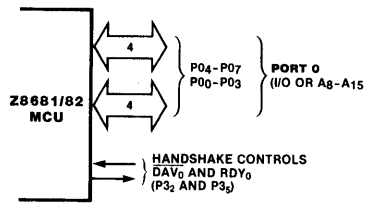


Figure 9. Port 0

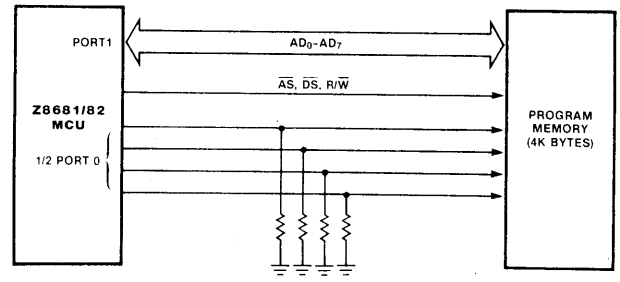


Figure 10. Port 0 Address Lines Tied to Logic 0

*This feature differs in the Z8681 and Z8682

I/O Ports (Continued)

Port 2 bits can be programmed independently as input or output (Figure 11). This port is always available for I/O operations. In addition, Port 2 can be configured to provide open-drain outputs.

Like Port 0, Port 2 may also be placed under handshake control. In this configuration, Port 3 lines $P3_1$ and $P3_6$ are used as the handshake controls lines DAV_2 and RDY_2. The handshake signal assignment for Port 3 lines $P3_1$ and $P3_6$ is dictated by the direction (input or output) assigned to bit 7 of Port 2.

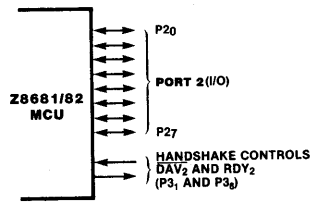


Figure 11. Port 2

Port 3 lines can be configured as I/O or control lines (Figure 12). In either case, the direction of the eight lines is fixed as four input ($P3_0$–$P3_3$) and four output ($P3_4$–$P3_7$). For serial I/O, lines $P3_0$ and $P3_7$ are programmed as serial in and serial out, respectively.

Port 3 can also provide the following control functions: handshake for Ports 0 and 2 ($\overline{DAV}$ and RDY); four external interrupt request signals (IRQ_0–IRQ_3); timer input and output signals (T_{IN} and T_{OUT}) and Data Memory Select ($\overline{DM}$).

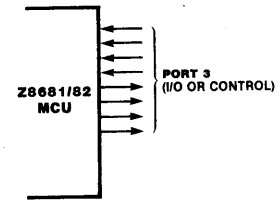


Figure 12. Port 3

Interrupts*

The Z8681/82 allows six different interrupts from eight sources: the four Port 3 lines $P3_0$–$P3_3$, Serial In, Serial Out, and the two counter/timers. These interrupts are both maskable and prioritized. The Interrupt Mask register globally or individually enables or disables the six interrupt requests. When more than one interrupt is pending, priorities are resolved by a programmable priority encoder that is controlled by the Interrupt Priority register.

All Z8681 and Z8682 interrupts are vectored through locations in program memory. When an interrupt request is granted, an interrupt machine cycle is entered. This disables all subsequent interrupts, saves the Program Counter and status flags, and accesses the program memory vector location reserved for that interrupt. In the Z8681, this memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request. The Z8681 takes 26 system clock cycles to enter an interrupt subroutine.

The Z8682 has a small internal ROM that contains six 2-byte interrupt vectors pointing to addresses 2048–2065, where 3-byte jump absolute instructions are located (See Figure 4). These jump instructions each contain a 1-byte opcode and a 2-byte starting address for the interrupt service routine. The Z8682 takes 36 system clock cycles to enter an interrupt subroutine.

Table 1. Z8682 Interrupt Processing

Address (Hex)	Contains Jump Instruction and Subroutine Address For
800-802	IRQ0
803-805	IRQ1
806-808	IRQ2
809-80B	IRQ3
80C-80E	IRQ4
80F-811	IRQ5

Polled interrupt systems are also supported. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Clock

The on-chip oscillator has a high-gain, parallel-resonant amplifier for connection to a crystal or to any suitable external clock source (XTAL1 = Input, XTAL2 = Output).

The crystal source is connected across XTAL1 and XTAL2, using the recommended capacitance (C_L = 15 pF maximum) from each pin to ground. The specifications for the crystal are as follows:

- AT cut, parallel-resonant
- Fundamental type
- Series resistance, $R_s \leq 100\Omega$
- For Z8681/Z8682, 8 MHz maximum
- For Z8681/Z8682-12, 12 MHz maximum

*This feature differs in the Z8681 and Z8682

Power Down Standby Option

The low-power standby mode allows power to be removed without losing the contents of the 124 general-purpose registers. This mode is available only to the user as a bonding option whereby pin 2 (normally XTAL2) is replaced by the V_{MM} (standby) power supply input. This necessitates the use of an external clock generator (input = XTAL1) rather than a crystal source.

The removal of power, whether intended or due to power failure, must be preceded by a software routine that stores the appropriate status into the register file. Figure 13 shows the recommended circuit for a battery back-up supply system.

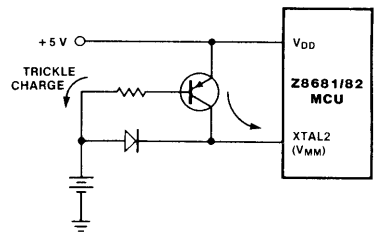


Figure 13. Recommended Driver Circuit for Power-Down Operation

Z8681/Z8682 Interchangeability

Although the Z8681 and Z8682 have minor differences, a system can be designed for compatibility with both ROMless versions. To achieve interchangeability, the design must take into account the special requirements of each device in the external interface, initialization, and memory mapping.

External Interface. The Z8682 requires a 7.5 V positive logic level on the $\overline{\text{RESET}}$ pin for at least 6 clock periods immediately following reset, as shown in Figure 14. The Z8681 requires a 3.8 V or higher positive logic level, but is compatible with the Z8682 $\overline{\text{RESET}}$ waveform. Figure 15 shows a simple circuit for generating the 7.5 V level.

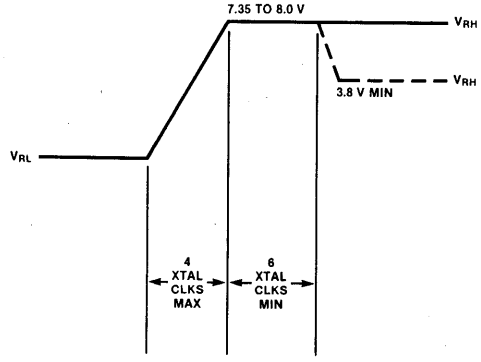


Figure 14. Z8682 $\overline{\text{RESET}}$ Pin Input Waveform

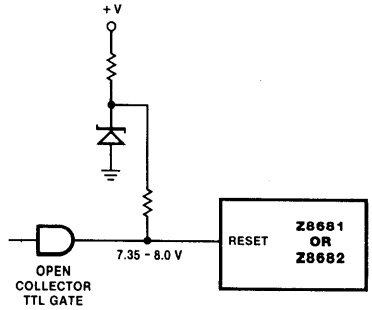


Figure 15. $\overline{\text{RESET}}$ Circuit

Initialization. The Z8681 wakes up after reset with Port 0 configured as an input, which means Port 0 lines are floating in a high-impedance state. Because of this pullup or pulldown, resistors must be attached to Port 0 lines to force them to a valid logic level until Port 0 is configured as an address port.

Port 0 initialization is discussed in the section on ports. An example of an initialization routine for Z8681/Z8682 compatibility is shown in Table 2. Only the Z8681 need execute this program.

Table 2. Initialization Routine

Address	Opcodes	Instruction	Comments
000C	E6 00 00	LD PO #%00	Set A_8-A_{15} to 0.
000F	E6 F8 96	LD P01M #%96	Configure Port 0 as A_8-A_{15}. Eliminate extended memory timing.
0012	8D 08 12	JP START ADDRESS	Execute application program.

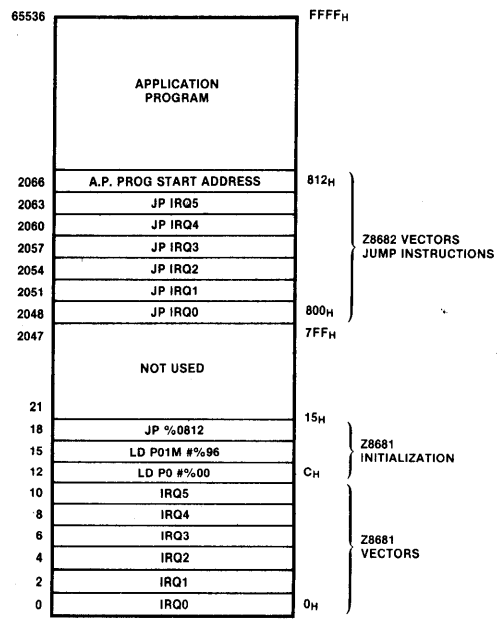
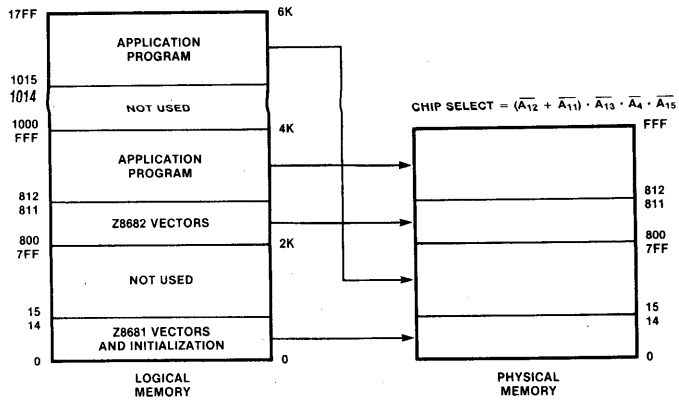


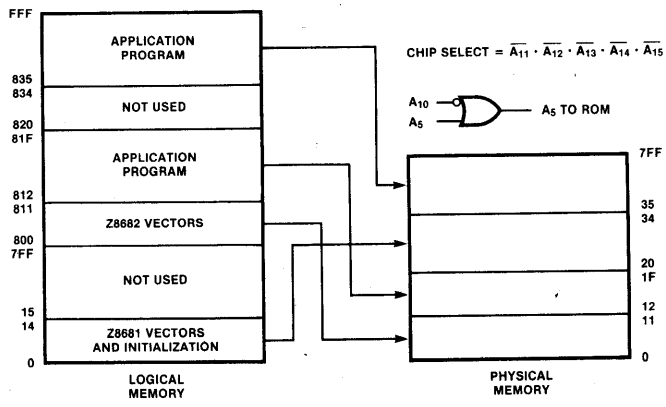
Figure 16. Z8681/82 Logical Program Memory Mapping

Z8681/Z8682 Interchangeability (Continued)

Memory Mapping. The Z8681 and Z8682 lower memory boundaries are located at 0 and 2048, respectively. A single program ROM can be used with either product if the logical program memory map shown in Figure 16 is followed. The Z8681 vectors and initialization routine must be starting at address 0 and the Z8682 3-byte vectors (jump instructions) must be at address 2048 and higher. Addresses in the range 21-2047 are not used. Figure 17 shows practical schemes for implementing this memory map using 4K and 2K ROMs.



a. Logical to Physical Memory Mapping for 4K ROM



b. Logical to Physical Memory Mapping for 2K ROM

Figure 17. Practical Schemes for Implementing Z8681 and Z8682 Compatible Memory Map

Instruction Set Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working register pair address

Symbols. The following symbols are used in describing the instruction set.

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack pointer (control registers 254-255)
PC	Program counter
FLAGS	Flag register (control register 252)
RP	Register pointer (control register 253)
IMR	Interrupt mask register (control register 251)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Instruction Set Notation (Continued)

Flags. Control Register R252 contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

Value	Mnemonic	Meaning	Flags Set
1000		Always true	---
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	---

Instruction Formats

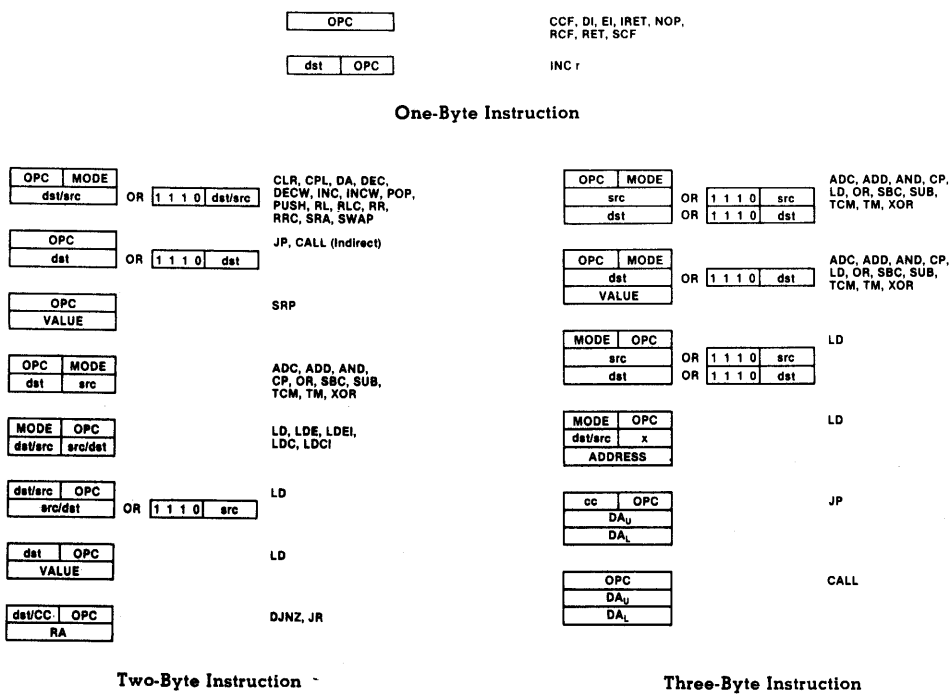
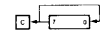
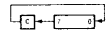
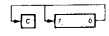
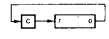
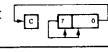


Figure 18. Instruction Formats

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r=0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r=0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c=0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c=0-F	- - - - - -
LD dst,src dst ← src	r r R r X X r Ir R R R IR R IM IR IM IR R	Im R r X r Ir r R IR IM IM R	rC r8 r9 r=0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst 	R IR		90 91	* * * * - -
RLC dst 	R IR		10 11	* * * * - -
RR dst 	R IR		E0 E1	* * * * - -
RRC dst 	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst 	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst 	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst, src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

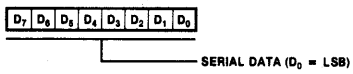
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

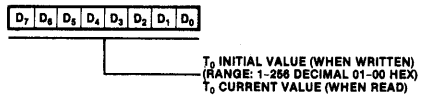
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

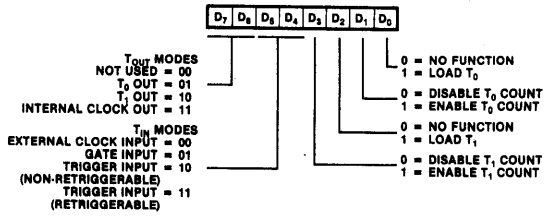
R240 SIO
Serial I/O Register
($F0_H$; Read/Write)



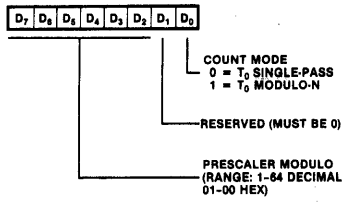
R244 TO
Counter/Timer 0 Register
($F4_H$; Read/Write)



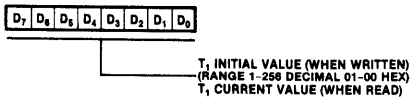
R241 TMR
Time Mode Register
($F1_H$; Read/Write)



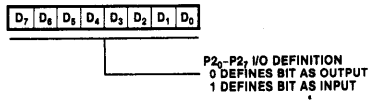
R245 PRE0
Prescaler 0 Register
($F5_H$; Write Only)



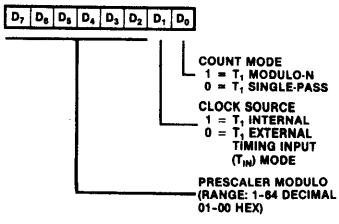
R242 T1
Counter Timer 1 Register
($F2_H$; Read/Write)



R246 P2M
Port 2 Mode Register
($F6_H$; Write Only)



R243 PRE1
Prescaler 1 Register
($F3_H$; Write Only)



R247 P3M
Port 3 Mode Register
($F7_H$; Write Only)

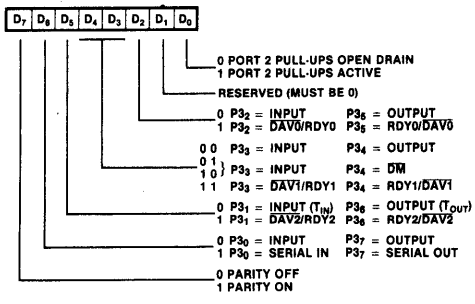
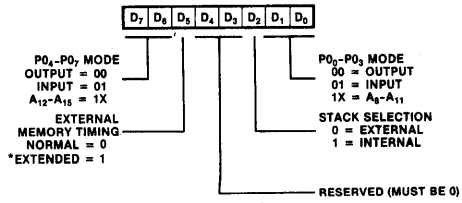


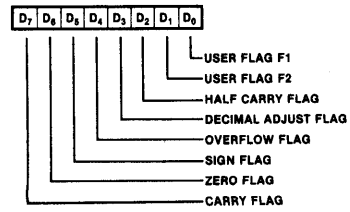
Figure 19. Control Registers

R248 P01M
Port 0 Register
($F8_H$; Write Only)

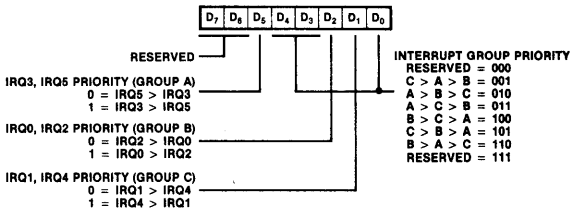


*ALWAYS EXTENDED TIMING AFTER RESET

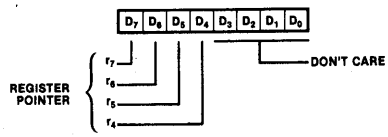
R252 FLAGS
Flag Register
(FC_H; Read/Write)



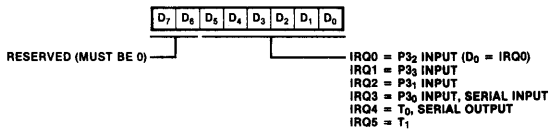
R249 IPR
Interrupt Priority Register
($F9_H$; Write Only)



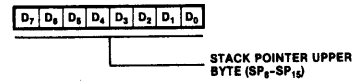
R253 RP
Register Pointer
(FD_H; Read/Write)



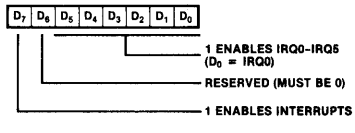
R250 IRQ
Interrupt Request Register
(FA_H; Read/Write)



R254 SPH
Stack Pointer
(FE_H; Read/Write)



R251 IMR
Interrupt Mask Register
(FB_H; Read/Write)



R255 SPL
Stack Pointer
(FF_H; Read/Write)

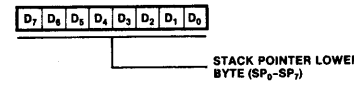
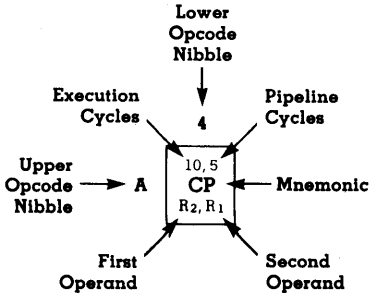


Figure 19. Control Registers (Continued)

Z8681/82 Opcode Map

Lower Nibble (Hex)

Upper Nibble (Hex)	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1, r_2	6,5 **ADD** r_1, Ir_2	10,5 **ADD** R_2, R_1	10,5 **ADD** IR_2, R_1	10,5 **ADD** R_1, IM	10,5 **ADD** IR_1, IM	6,5 **LD** r_1, R_2	6,5 **LD** r_2, R_1	12/10,5 **DJNZ** r_1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r_1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r_1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1, r_2	6,5 **ADC** r_1, Ir_2	10,5 **ADC** R_2, R_1	10,5 **ADC** IR_2, R_1	10,5 **ADC** R_1, IM	10,5 **ADC** IR_1, IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1, r_2	6,5 **SUB** r_1, Ir_2	10,5 **SUB** R_2, R_1	10,5 **SUB** IR_2, R_1	10,5 **SUB** R_1, IM	10,5 **SUB** IR_1, IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1, r_2	6,5 **SBC** r_1, Ir_2	10,5 **SBC** R_2, R_1	10,5 **SBC** IR_2, R_1	10,5 **SBC** R_1, IM	10,5 **SBC** IR_1, IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1, r_2	6,5 **OR** r_1, Ir_2	10,5 **OR** R_2, R_1	10,5 **OR** IR_2, R_1	10,5 **OR** R_1, IM	10,5 **OR** IR_1, IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1, r_2	6,5 **AND** r_1, Ir_2	10,5 **AND** R_2, R_1	10,5 **AND** IR_2, R_1	10,5 **AND** R_1, IM	10,5 **AND** IR_1, IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1, r_2	6,5 **TCM** r_1, Ir_2	10,5 **TCM** R_2, R_1	10,5 **TCM** IR_2, R_1	10,5 **TCM** R_1, IM	10,5 **TCM** IR_1, IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1, r_2	6,5 **TM** r_1, Ir_2	10,5 **TM** R_2, R_1	10,5 **TM** IR_2, R_1	10,5 **TM** R_1, IM	10,5 **TM** IR_1, IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1, Irr_2	18,0 **LDEI** Ir_1, Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2, Irr_1	18,0 **LDEI** Ir_2, Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1, r_2	6,5 **CP** r_1, Ir_2	10,5 **CP** R_2, R_1	10,5 **CP** IR_2, R_1	10,5 **CP** R_1, IM	10,5 **CP** IR_1, IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1, r_2	6,5 **XOR** r_1, Ir_2	10,5 **XOR** R_2, R_1	10,5 **XOR** IR_2, R_1	10,5 **XOR** R_1, IM	10,5 **XOR** IR_1, IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1, Irr_2	18,0 **LDCI** Ir_1, Irr_2				10,5 **LD** r_1, x, R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2, Irr_1	18,0 **LDCI** Ir_2, Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2, x, R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1, IR_2	10,5 **LD** R_2, R_1	10,5 **LD** IR_2, R_1	10,5 **LD** R_1, IM	10,5 **LD** IR_1, IM								6,5 **CCF**
F	8,5 **SWAP** R_1	8,5 **SWAP** IR_1		6,5 **LD** Ir_1, r_2		10,5 **LD** R_2, IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



Legend:
R = 8-Bit Address
r = 4-Bit Address
R_1 or r_1 = Dst Address
R_2 or r_2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction

Absolute Maximum Ratings

Voltages on all pins* with respect to GND . . . . . -0.3 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. *Positive current flows into the reference* pin. Standard conditions are as follows:

- +4.75 V ≤ V_{CC} ≤ +5.25 V
- GND = 0 V
- 0°C ≤ T_A ≤ +70°C for S (Standard temperature)
- -40°C ≤ T_A ≤ +85°C for E (Extended temperature)

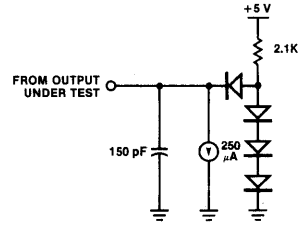


Figure 20. Test Load 1

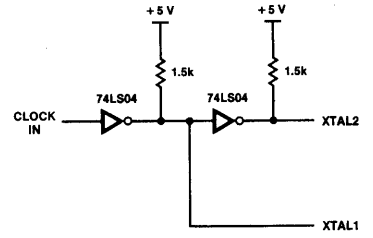


Figure 21. External Clock Interface Circuit

DC Characteristics

Symbol	Parameter	Min	Max	Unit	Condition
V_{CH}	Clock Input High Voltage	3.8	V_{CC}	V	Driven by External Clock Generator
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V	Driven by External Clock Generator
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{RH}	Reset Input High Voltage	3.8	V_{CC}	V	See Note
V_{RL}	Reset Input Low Voltage	-0.3	0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA
I_{IL}	Input Leakage	-10	10	µA	0 V ≤ V_{IN} ≤ +5.25 V
I_{OL}	Output Leakage	-10	10	µA	0 V ≤ V_{IN} ≤ +5.25 V
I_{IR}	Reset Input Current		-50	µA	V_{CC} = +5.25 V, V_{RL} = 0 V
I_{CC}	V_{CC} Supply Current		180	mA	
I_{MM}	V_{MM} Supply Current		10	mA	Power Down Mode
V_{MM}	Backup Supply Voltage	3	V_{CC}	V	Power Down

NOTE:
The Reset line (pin 6) is used to place the Z8682 in external memory mode. This is accomplished as shown in Figure 14.

*Except RESET Pin 6

External I/O or Memory Read and Write Timing

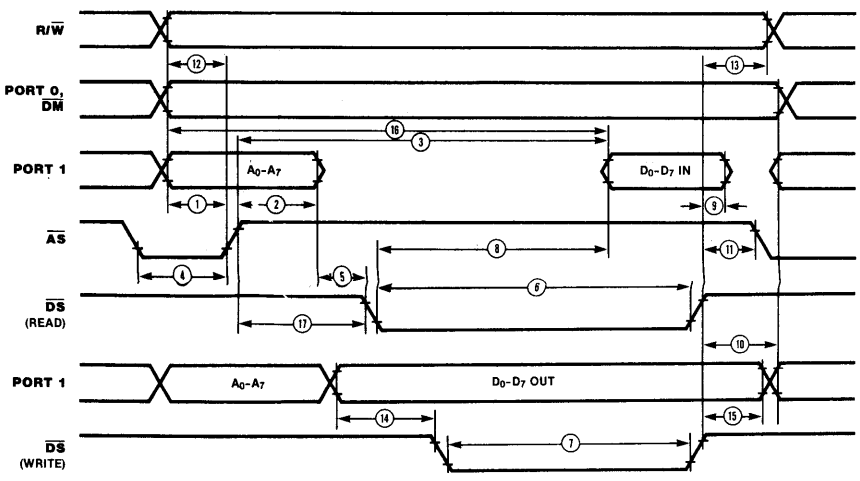


Figure 22. External I/O or Memory Read/Write Timing

No.	Symbol	Parameter	Z8681/82 8 MHz Min	Z8681/82 8 MHz Max	Z8681/82 12 MHz Min	Z8681/82 12 MHz Max	Notes*†
1	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		1,2,3
2	TdAS(A)	$\overline{AS}$ ↑ to Address Float Delay	70		45		1,2,3
3	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220	1,2,3,4
4	TwAS	$\overline{AS}$ Low Width	80		55		1,2,3
5	TdAz(DS)	Address Float to $\overline{DS}$ ↓	0		0		1
6	TwDSR	$\overline{DS}$ (Read) Low Width	250		185		1,2,3,4
7	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		1,2,3,4
8	TdDSR(DR)	$\overline{DS}$ ↓ to Read Data Required Valid		200		130	1,2,3,4
9	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		1
10	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	70		45		1,2,3
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		55		1,2,3
12	TdR/W(AS)	R/$\overline{W}$ Valid to $\overline{AS}$ ↑ Delay	50		30		1,2,3
13	TdDS(R/W)	$\overline{DS}$ ↑ to R/$\overline{W}$ Not Valid	60		35		1,2,3
14	TdDW(DSW)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		1,2,3
15	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	70		45		1,2,3
16	TdA(DR)	Address Valid to Read Data Required Valid		410		255	1,2,3,4
17	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80		55		1,2,3

NOTES:
1. Test Load 1
2. Timing numbers given are for minimum TpC.
3. Also see clock cycle time dependent characteristics table.
4. When using extended memory timing add 2 TpC.
5. All timing references use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* All units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Additional Timing Table

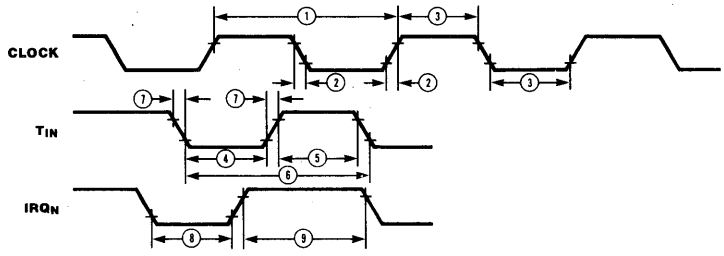


Figure 23. Additional Timing

No.	Symbol	Parameter	Z8681/82 8 MHz Min	Z8681/82 8 MHz Max	Z8681/82 12 MHz Min	Z8681/82 12 MHz Max	Notes*‡
1	TpC	Input Clock Period	125	1000	83	1000	1
2	TrC,TfC	Clock Input Rise And Fall Times		25		15	1
3	TwC	Input Clock Width	37		26		1
4	TwTinL	Timer Input Low Width	100		70		2
5	TwTinH	Timer Input High Width	3TpC		3TpC		2
6	TpTin	Timer Input Period	$\frac{TpC}{8}$		$\frac{TpC}{8}$		2
7	TrTin,TfTin	Timer Input Rise And Fall Times		100		100	2
8	TwIL	Interrupt Request Input Low Time	100		70		2,3
9	TwIH	Interrupt Request Input High Time	3TpC		3TpC		2,3

NOTES:
1. Clock timing references uses 3.8 V for a logic "1" and 0.8 V for a logic "0".
2. Timing reference uses 2.0 V for a logic "1" and 0.8 V for a logic "0".
3. Interrupt request via Port 3.
* Units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Handshake Timing

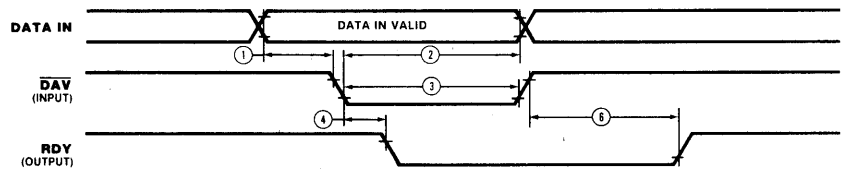


Figure 24a. Input Handshake Timing

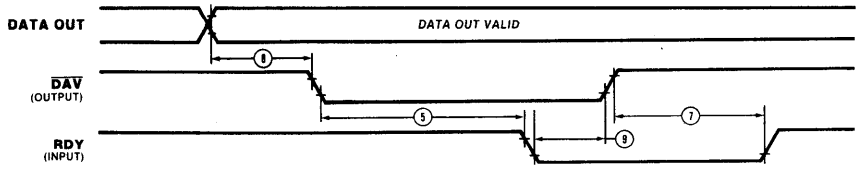


Figure 24b. Output Handshake Timing

No.	Symbol	Parameter	Z8681/82 8 MHz Min	Z8681/82 8 MHz Max	Z8681/82 12 MHz Min	Z8681/82 12 MHz Max	Notes*†
1	TsDI(DAV)	Data In Setup Time	0		0		
2	ThDI(DAV)	Data In Hold Time	230		160		
3	TwDAV	Data Available Width	175		120		
4	TdDAVIf(RDY)	$\overline{DAV}$ ↓ Input to RDY ↓ Delay		175		120	1,2
5	TdDAVOf(RDY)	$\overline{DAV}$ ↓ Output to RDY ↓ Delay	0		0		1,3
6	TdDAVIr(RDY)	$\overline{DAV}$ ↑ Input to RDY ↑ Delay		175		120	1,2
7	TdDAV0rRDY)	$\overline{DAV}$ ↑ Output to RDY ↑ Delay	0		0		1,3
8	TdDO(DAV)	Data Out to $\overline{DAV}$ ↓ Delay	50		30		1
9	TdRDY(DAV)	Rdy ↓ Input to $\overline{DAV}$ ↑ Delay	0	200	0	140	1

NOTES:
1. Test load 1
2. Input handshake
3. Output handshake
4. All timing regerences use 2.0 V for a logic "1" and 0.8 V for a logic "0".

* Units in nanoseconds (ns).
† Timings are preliminary and subject to change.

Clock-Cycle-Time-Dependent Characteristics

Number	Symbol	Z8681/82 8 MHz Equation	Z8681/82 12 MHz Equation
1	TdA(AS)	TpC-75	TpC-50
2	TdAS(A)	TpC-55	TpC-40
3	TdAS(DR)	4TpC-140*	4TpC-110*
4	TwAS	TpC-45	TpC-30
6	TwDSR	3TpC-125*	3TpC-65*
7	TwDSW	2TpC-90*	2TpC-55*
8	TdDSR(DR)	3TpC-175*	3TpC-120*
10	Td(DS)A	TpC-55	TpC-40
11	TdDS(AS)	TpC-55	TpC-30
12	TdR/W(AS)	TpC-75	TpC-55
13	TdDS(R/W)	TpC-65	TpC-50
14	TdDW(DSW)	TpC-75	TpC-50
15	TdDS(DW)	TpC-55	TpC-40
16	TdA(DR)	5TpC-215*	5TpC-160*
17	TdAS(DS)	TpC-45	TpC-30

* Add 2TpC when using extended memory timing

Ordering Information

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8681	CE	8.0 MHz	Z8 MCU (ROMless, 40-pin)	Z8681	CE	12.0 MHz	Z8 MCU (ROMless, 40-pin)
Z8681	CS	8.0 MHz	Same as above	Z8681	CS	12.0 MHz	Same as above
Z8681	DE	8.0 MHz	Same as above	Z8681	DE	12.0 MHz	Same as above
Z8681	DS	8.0 MHz	Same as above	Z8681	DS	12.0 MHz	Same as above
Z8681	PE	8.0 MHz	Same as above	Z8681	PE	12.0 MHz	Same as above
Z8681	PS	8.0 MHz	Same as above	Z8681	PS	12.0 MHz	Same as above
Z8682	PE	8.0 MHz	Same as above	Z8682	PE	12.0 MHz	Same as above
Z8682	PS	8.0 MHz	Same as above	Z8682	PS	12.0 MHz	Same as above

NOTES: C = Ceramic, D = Cerdip, P = Plastic; E = -40°C to +85°C, S = 0°C to +70°C.
NOTE: The Z8681 is available in the low power standby option. If this option is desired, a Z8685 part number should be specified. All other data remains the same for ordering purposes.

Z8® Family of Universal Peripheral Controllers

Product Specification

Zilog

September 1983

Non-Multiplexed Bus
Z8590—2K ROM
Z8594—PROM/RAM Protopack
Z-BUS
Z8090—2K ROM
Z8094—PROM/RAM Protopack

Features

- Complete slave Z8 microcomputer, for distributed processing use.
- Unmatched power of Z8 architecture and instruction set.
- Three programmable I/O ports, two with optional 2-Wire Handshake.
- Six levels of priority interrupts from eight sources: six from external sources and two from internal sources.
- Two programmable 8-bit counter/timers each with a 6-bit prescaler. Counter/Timer T0 is driven by an internal source, and Counter/Timer T1 can be driven by internal or external sources. Both counter/timers are independent of program execution.
- 256-byte register file, accessible by both the master CPU and UPC, as allocated in the UPC program.
- 2K bytes of on-chip ROM for efficiency and versatility.

General Description

The Universal Peripheral Controller (UPC) is an intelligent peripheral controller for distributed processing applications (Figure 1). The UPC unburdens the host processor by assuming tasks traditionally done by the host (or by added hardware), such as performing arithmetic, translating or formatting data, and controlling I/O devices. Based on the Z8 microcomputer architecture and instruction set, the UPC contains 2K bytes of internal program ROM, a 256-byte register file, three 8-bit I/O ports, and two counter/timers.

The UPC is offered in two basic configurations: the Z8090/4, which interfaces to multiplexed address/data CPUs such as the Z8000, and the Z8590/4, which interfaces with non-multiplexed CPUs such as the Z80. Both devices have the same instruction set and I/O port configuration. The difference in the devices is in the UPC-to-host interface pins and the sequence of data transfer between the units.

The UPC offers fast execution time, an effective use of memory, and sophisticated interrupt, I/O, and bit manipulation. Using a powerful and extensive instruction set combined with an efficient internal addressing scheme, the UPC speeds program execution and efficiently packs program code into the on-chip ROM.

General Description (Continued)

An important feature of the UPC is an internal register file containing I/O port and control registers accessed both by the UPC program and indirectly by its associated master CPU. This architecture results in both byte and programming efficiency, because UPC instructions can operate directly on I/O data without moving it to and from an accumulator. Such a structure allows the user to allocate as many general-purpose registers as the application requires for data buffers between the CPU and peripheral devices. All general-purpose registers can be used as address pointers, index registers, data buffers, or stack space.

The register file is logically divided into 16 groups, each consisting of 16 working registers. A Register Pointer is used in conjunction with short format instructions, resulting in tight, fast code and easy task switching.

Communication between the master CPU and the register file takes place via one group of 19 interface registers addressed directly by both the master CPU and the UPC, or via a block transfer mechanism. Access by the master CPU is controlled by the UPC to allow independence between the master CPU and UPC software.

The UPC has 24 pins that can be dedicated to I/O functions. Grouped logically into three 8-line ports, they can be programmed in many combinations of input or output lines, with or without handshake, and with push-pull or open-drain outputs. Ports 1 and 2 are bit-programmable; Port 3 has four fixed inputs and four outputs.

To relieve software from coping with real-time counting and timing problems, the UPC has two 8-bit hardware counter/timers, each with a fixed divide-by-four, and a 6-bit programmable prescaler. Various counting modes may be selected.

In addition to the 40-pin standard ROM configuration, the UPC is available in a Protopack RAM/ROM version with a socket for up to 2K bytes of RAM or ROM and with 36 bytes of internal ROM permitting downloading from the master CPU.

This range of versions and configurations makes the UPC compatible with most system peripheral device control considerations.

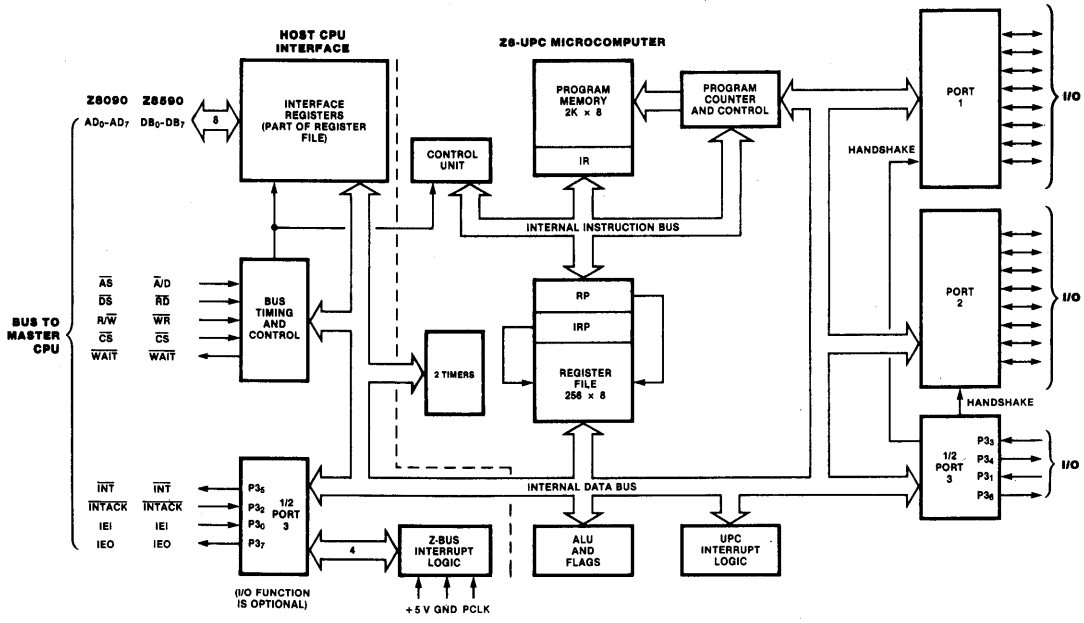


Figure 1. Functional Block Diagram

Pin Description Z8090 Z-UPC

AD_0-AD_7. *Z-Bus Address/Data Lines* (bidirectional). These multiplexed address and data lines are used to transfer information between the master CPU and the slave Z-UPC.

$\overline{AS}$. *Address Strobe* (input, active Low). The rising edge of $\overline{AS}$ initiates the beginning of a transaction and indicates that the Address, Status, R/$\overline{W}$, and $\overline{CS}$ signals must be valid.

PCLK. *Clock* (input). TTL-compatible clock input, 4 MHz maximum. This signal does not need to be related to the master CPU clock.

$\overline{CS}$. *Chip Select* (input, active Low). A Low on this line during the rising edge of $\overline{AS}$ enables the Z-UPC to accept address or data information from the bus during a master CPU write cycle or to transmit data to the bus during a read cycle.

$\overline{DS}$. *Data Strobe* (input, active Low). $\overline{DS}$ provides timing for data movement to the bus master. A simultaneous Low on $\overline{AS}$ and $\overline{DS}$ resets the Z-UPC. It is held in reset as long as $\overline{DS}$ is Low.

$P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (inputs/outputs, TTL-compatible). These 24 lines are divided into three 8-bit I/O ports and may be configured in the following ways under program control:

$P1_0$-$P1_7$. *Port 1* (input/output—as output it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P2_0$-$P2_7$. *Port 2* (input/output—as output, it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P3_0$-$P3_7$. *Port 3* (four inputs, four outputs). Parallel I/O, handshake control, timer I/O, or interrupt control.

R/$\overline{W}$. *Read/Write* (input). This status signal indicates that the master CPU is executing a Read cycle if High, and a Write cycle if Low.

$\overline{WAIT}$. *Wait* (output, active Low, open-drain). When the CPU accesses the Z-UPC register file, this signal requests the master CPU to wait until the Z-UPC can complete its part of the transaction.

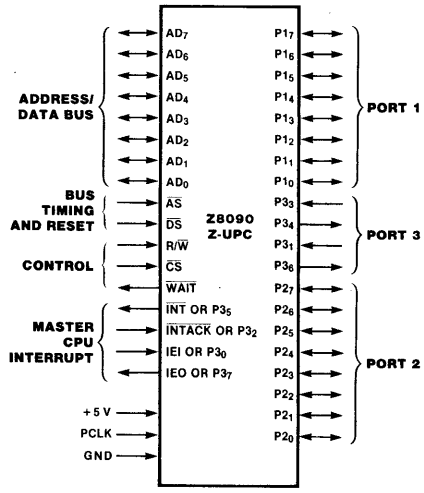


Figure 2. Pin Functions

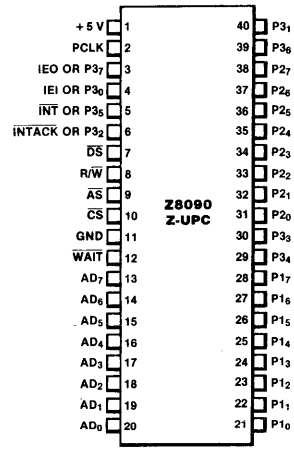


Figure 3. Pin Assignments

Pin Description Z8590 UPC

A/D. *Address/Data* (input). A Low on this pin defines information on the data bus as an address. A High defines the information as data.

$\overline{CS}$. *Chip Select* (input, active Low). A Low enables the UPC to accept address or data information from the master CPU during a write cycle or to transmit data to the master CPU during a read cycle. This line is usually generated from higher bits of the address lines.

DB_0-DB_7. *Data Bus* (bidirectional). This bus is used to transfer address and data information between the master CPU and the UPC.

$P1_0$-$P1_7$, $P2_0$-$P2_7$, $P3_0$-$P3_7$. *I/O Port Lines* (bidirectional, TTL-compatible). These 24 lines are divided into three 8-bit I/O ports and may be configured in the following ways under program control:

$P1_0$-$P1_7$. *Port 1* (input/output—as output it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P2_0$-$P2_7$. *Port 2* (input/output—as output, it can be push-pull or open-drain). Bit-programmable Parallel I/O.

$P3_0$-$P3_7$. *Port 3* (four inputs, four outputs). Parallel I/O, handshake control, timer I/O, or interrupt control.

PCLK. *Clock* (input). TTL-compatible clock input, 4 MHz maximum. This signal does not need to be related to the master CPU clock.

$\overline{RD}$. *Read* (input, active Low). A Low enables the master CPU to read information from the UPC. Raising the voltage on this pin above V_{DD} will force the UPC into test mode.

$\overline{WAIT}$. *Wait* (output, active Low, open-drain). When the CPU accesses the UPC register file, this signal requests the master CPU to wait until the UPC can complete its part of the transaction.

$\overline{WR}$. *Write* (input, active Low). A Low on this pin enables the master CPU to write information to the UPC. A simultaneous Low on $\overline{RD}$ and $\overline{WR}$ resets the UPC. It is held in reset as long as $\overline{WR}$ is Low.

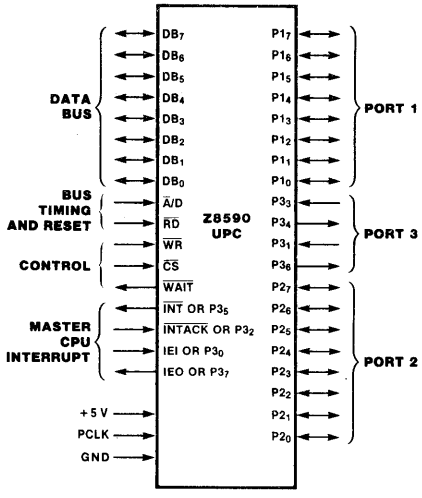


Figure 4. Z8590 UPC Pin Functions

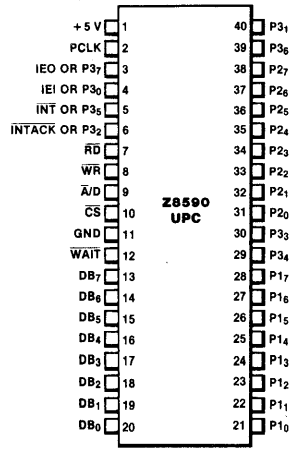


Figure 5. Z8590 UPC Pin Assignments

Functional Description

Address Space. On the 40-pin UPC, all address space is committed to on-chip memory. There are 2048 bytes of mask-programmed ROM and 256 bytes of register file. I/O is memory-mapped to three registers in the register file. Only the Protopack version of the UPC can access external program memory. See the section entitled "Special Configurations" for a complete description of the Protopack version.

Program Memory. Figure 6 is a map of the 2K on-chip program ROM. Even though the architecture allows addresses from 0 to 4K, behavior of the device above program address 2047 (7FFH) is not defined. The first 12 bytes of program memory are reserved for the UPC interrupt vectors. In the RAM version, addresses 0CH through 2FH are reserved for on-chip ROM.

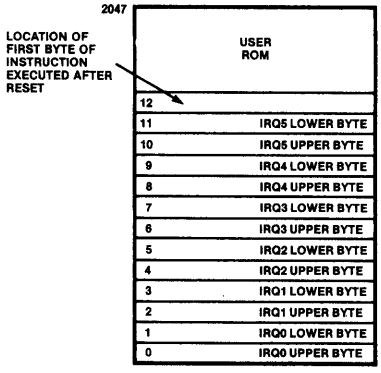


Figure 6. Program Memory Map

Register File. This 256-byte file includes three I/O port registers (1-3H), 234 general-purpose registers (6-EFH), and 19 control, status and special I/O registers (0H, 4H, 5H, and F0-FFH). The functions and mnemonics assigned to these register address locations are shown in Figure 7. Of the 256 UPC registers, 19 can be directly accessed by the master CPU; the others are accessed indirectly via the block transfer mechanism.

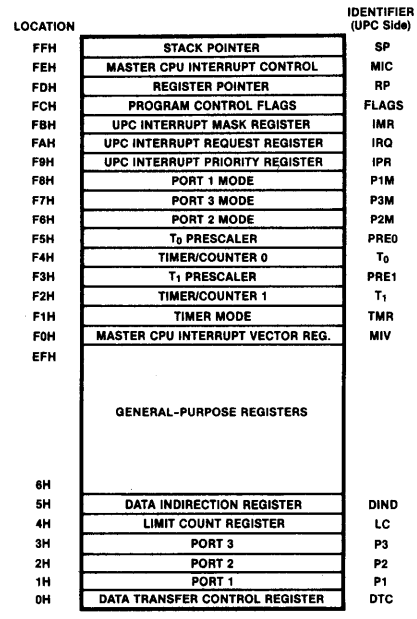


Figure 7. Register File Organization

Functional Description (Continued)

The I/O port and control registers are included in the register file without differentiation. This allows any UPC instruction to process I/O or control information, thereby eliminating the need for special I/O and control instructions. All general-purpose registers can function as accumulators, address pointers, or index registers. In instruction execution, the registers are read when they are defined as sources and written when defined as destinations.

UPC instructions may access registers directly or indirectly using an 8-bit address mode or a 4-bit address mode and a Register Pointer. For the 4-bit addressing mode, the file is divided into 16 working register groups, each occupying 16 contiguous locations (Figure 8). The Register Pointer (RP) addresses the starting point of the active working-register group, and the 4-bit register designator supplied by the instruction specifies the register within the group. Any instruction altering the contents of the register file can also alter the Register Pointer. The UPC instruction set has a special Set Register Pointer (SRP) instruction for initializing or altering the pointer contents.

Stacks. An 8-bit Stack Pointer (SP), register R255, is used for addressing the stack, residing within the 234 general-purpose registers, address location 6H through EFH. PUSH and POP instructions can save and restore any register in the register file on the stack. During CALL instructions, the Program Counter is automatically saved on the stack. During UPC interrupt cycles, the Program Counter and the Flag register are automatically saved on the stack. The RET and IRET instructions pop the saved values of the Program Counter and Flag register.

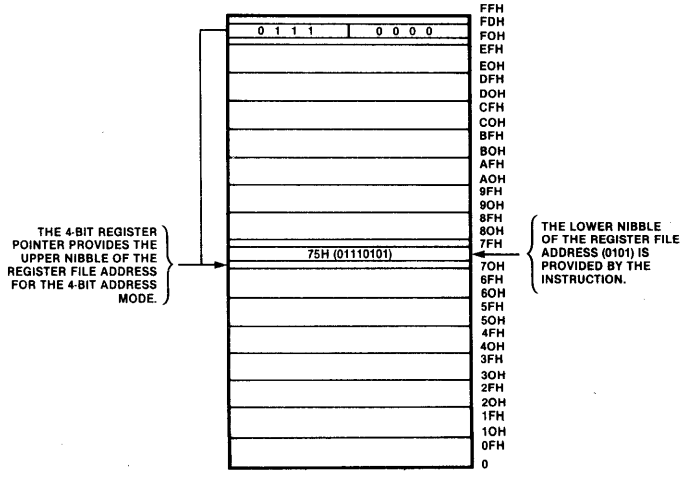


Figure 8. Register Pointer Mechanism

Ports. The UPC has 24 lines dedicated to input and output. These are grouped into three ports of eight lines each and can be configured under software control as inputs, outputs, or special control signals. They can be programmed to provide Parallel I/O with or without handshake and timing signals. All outputs can have active pullups and pulldowns, compatible with TTL loads. In addition, Port 1 and 2 may be configured as open-drain outputs.

Port 1. Individual bits of Port 1 can be configured as input or output by programming Port 1 Mode register (P1M) F8H. This port is accessed by the UPC program as general register 1H. It is written by specifying address 1H as the destination of any instruction used to store data in the output register. The port is read by specifying address 1H as the source of an instruction.

Port 1 may be placed under handshake control by programming Port 3 Mode register (P3M) F7H. This configures Port 3 pins $P3_3$ and $P3_4$ as handshake control lines $\overline{DAV}_1$ and RDY_1 for input handshake, or RDY_1 and $\overline{DAV}_1$ for output handshake, as determined by the direction (input or output) assigned to bit 7 of Port 1. The Port 3 Mode register also has a bit that programs Port 1 for open-drain output.

Port 2. Individual bits of Port 2 can be configured as inputs or outputs by programming Port 2 Mode register (P2M) F6H. This port is accessed by the UPC program as general register 2H, and its functions and methods of programming are the same as those of Port 1. Port 3 pins $P3_1$ and $P3_6$ are the handshake lines $\overline{DAV}_2$ and RDY_2, with the direction (input or output) determined by the state of bit 7 of the port. The Port 3 Mode register also has a bit used to program Port 2 for open-drain output.

Function	Line	Direction	Signal
Handshake	$P3_1$	In	$\overline{DAV}_2/RDY_2$
	$P3_3$	In	$\overline{DAV}_1/RDY_1$
	$P3_4$	Out	$RDY_1/\overline{DAV}_1$
	$P3_6$	Out	$RDY_2/\overline{DAV}_2$
UPC Interrupt Request*	$P3_0$	In	IRQ_3
	$P3_1$	In	IRQ_2
	$P3_3$	In	IRQ_1
Counter/Timer	$P3_1$	In	T_{7N}
	$P3_6$	Out	T_{OUT}
Master CPU	$P3_5$	Out	$\overline{INT}$
	$P3_2$	In	$\overline{INTACK}$
	$P3_0$	In	IEI
	$P3_7$	Out	IEO
Test Mode	$P3_5$	Out	$\overline{A}/D$

*$P3_0$, $P3_1$, and $P3_3$ can always be used as UPC interrupt request inputs, regardless of the configuration programmed.

Table 1. Port 3 Control Functions

Functional Description (Continued)

Port 3. This port can be configured as I/O or control lines by programming the Port 3 Mode register. Port 3 is accessed as general register 3H. The directions of the eight data lines are fixed. Four lines, $P3_0$ through $P3_3$, are inputs, and the other four, $P3_4$ through $P3_7$, are outputs. The control functions performed by Port 3 are listed in Table 1

Counter/Timers. The UPC contains two 8-bit programmable counter/timers, each driven by an internal 6-bit programmable prescaler.

The T1 prescaler can be driven by internal or external clock sources. The T0 prescaler is driven by an internal clock source. Both counter/timers operate independently of the processor instruction sequence to relieve the program from time-critical operations like event counting or elapsed-time calculation. T0 Prescaler register (PRE0) F5H and T1 Prescaler register (PRE1) F3H can be programmed to divide the input frequency of the source being counted by any number from 1 to 64. A Counter register (F2H or F4H) is loaded with a number from 1 to 256. The corresponding counter is decremented from this number each time the prescaler reaches end-of-count. When the count is complete, the counter issues a timer interrupt request; IRQ_4 for T0 or IRQ_5 for T1. Loading either counter with a number (n) results in the interruption of the UPC at the nth count.

The counters can be started, stopped, restarted to continue, or restarted from the initial value. They can be programmed to stop upon reaching end-of-count (Single-Pass mode) or to automatically reload the initial value and continue counting (Modulo-n Continuous mode). The counters and prescalers can be read at any time without disturbing their values or changing their counts. The clock source for timer T1 can be defined as any one of the following:

- UPC internal clock (4 MHz maximum) divided by four.
- External clock input to Counter/Timer T1 via $P3_1$ (1 MHz maximum).
- Retriggerable trigger input for the UPC internal clock divided by four.
- Nonretriggerable trigger input for the UPC internal clock divided by four.
- External gate input for the UPC internal clock divided by four.

T0 is driven by the UPC internal clock divided by four.

Interrupts. The UPC allows six interrupts from eight different sources as follows:

- Port 3 lines $P3_0$, $P3_2$, and $P3_3$.
- The master CPU(3).
- The two counter/timers.

These interrupts can be masked and globally enabled or disabled using Interrupt Mask Register (IMR) FBH. Interrupt Priority Register (IPR) F9H specifies the order of their priority. All UPC interrupts are vectored.

Table 2 lists the UPC's interrupt sources, their types, and their vector locations in program ROM. Interrupt Request IRQ_0 is dedicated to master CPU communications. Interrupt Requests IRQ_1, IRQ_2, and IRQ_3 are generated on the falling transitions of external inputs $P3_3$, $P3_1$, and $P3_0$. Interrupt Requests IRQ_4 and IRQ_5 are generated upon the timeout of the UPC's two counter/timers. When an interrupt request is granted, the UPC enters an interrupt machine cycle. This cycle disables all subsequent interrupts, saves the Program Counter and Status Flags, and branches to the program memory vector location reserved for that interrupt. This memory location and the next byte contain the 16-bit address of the interrupt service routine for that particular interrupt request.

The UPC also supports polled systems. To accommodate a polled structure, any or all of the interrupt inputs can be masked and the Interrupt Request register polled to determine which of the interrupt requests needs service.

Following any hardware reset operation, an EI instruction must be executed to enable the setting of any interrupt request bit in the IRQ register. Interrupts must be disabled prior to changing the content of either the IPR (F9H) or the IMR (FBH). DI is the only instruction that should be used to globally disable interrupts.

Name	Source	Vector Location	Comments
IRQ_0	EOM, XERR, LERR	0,1	Internal (R0 Bits 0, 1, 2)
IRQ_1	$\overline{DAV}_1$, IRQ_1	2,3	External ($P3_3$) ↓ Edge Triggered
IRQ_2	$\overline{DAV}_2$, IRQ_2, T_{IN}	4,5	External ($P3_1$) ↓ Edge Triggered
IRQ_3	IRQ_3, IEI	6,7	External ($P3_0$) ↓ Edge Triggered
IRQ_4	T0	8,9	Internal
IRQ_5	T1	10,11	Internal

Table 2. Interrupt Types, Sources, and Vector Locations

Functional Description (Continued)

Master CPU Register File Access. There are two ways in which the master CPU can access the UPC register file: direct access and block access.

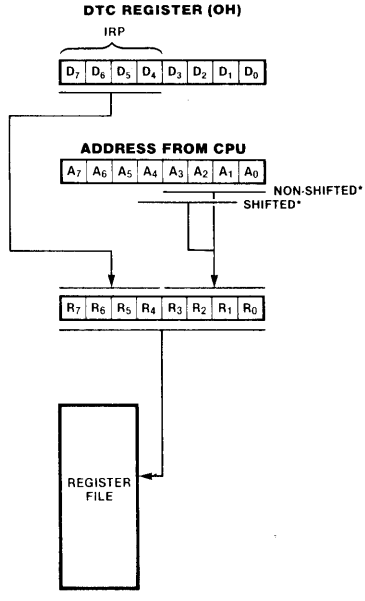
Direct Access. Three UPC registers—the Data Transfer Control (0H), the Master Interrupt Vector (F0H), and the Master Interrupt Control (FEH)—are mapped directly into the master CPU address space. The master CPU accesses these registers via the addresses shown in Table 3.

The master CPU also has direct access to 16 registers known as the DSC (Data, Status, Command) registers. The DSC registers are numbered 0 through F (DSC0-DSCF). These registers can be any 16 contiguous register file registers beginning on a 16-byte boundary. The base address of the DSC register group is designated by the IRP (I/O Register Pointer), which is bits D_4-D_7 of the Data Transfer Control register (0H). Figure 9 shows how the register address is made up of the 4-bit IRP field, concatenated with the low order 4-bits of the address from the master CPU.

Block Access. The master CPU may transmit or receive blocks of data via address xxx10101. When the master CPU accesses this address, the UPC register pointed to by the Data Indirection register is read or written. The Data Indirection register is incremented, and the Limit Count register is decremented, for example, when the master CPU issues a read or write to address xxx10101 while the Data Indirection register contains the value 33H. The operation causes register 33H to be read or written and the Data Indirection register to be incremented to 34H. This scheme is well suited to Block I/O Instructions and allows the master CPU to efficiently read or write a block of data to or from the UPC.

The Limit Count register (04H) is decremented and is used to control the number of bytes to be transferred by master CPU block accesses. If the master CPU attempts a read or write to the UPC after the Limit Count register reaches 0, the access is not completed, the LERR bit (D_1) of the Data Transfer Control register is set (indicating a limit error), and the LERR error causes an IRQ_0 interrupt request.

The IRP field of the Data Transfer Control register, the Data Indirection register, and the Limit Count register are not directly accessible to the master CPU and therefore must be set by the UPC. This allows the UPC to protect itself from master CPU errors and frees the master CPU from tracking the UPC's internal data layout.



*The shift or no-shift state is set during a hardware reset. If the Wait line is held High during the hardware reset, the 8090/4 is in the shift state after the reset. If $\overline{WAIT}$ is held Low, it is in the no-shift state. The shift state is maintained until the next hardware reset. Figure 7 shows one way to interface the 8090/4 for the use of no-shift.

Figure 9. DCS Register Addressing Scheme

UPC Address Decimal	UPC Address Hex	Identifier	8090/4 8590/4 No-Shift Address	8090/4 Shift Address
0	0H	DTC	xxx11000	xx11000x
5	5H	DIND		
@5**	@5H**		xxx10101	xx10101x
240	F0H	MIV	xxx10000	xx10000x
254	FEH	MIC	xxx11110	xx11110x
*n		DSC0	xxx00000	xx00000x
n+1		DSC1	xxx00001	xx00001x
n+2		DSC2	xxx00010	xx00010x
n+3		DSC3	xxx00011	xx00011x
n+4		DSC4	xxx00100	xx00100x
n+5		DSC5	xxx00101	xx00101x
n+6		DSC6	xxx00110	xx00110x
n+7		DSC7	xxx00111	xx00111x
n+8		DSC8	xxx01000	xx01000x
n+9		DSC9	xxx01001	xx01001x
n+10		DSCA	xxx01010	xx01010x
n+11		DSCB	xxx01011	xx01011x
n+12		DSCC	xxx01100	xx01100x
n+13		DSCD	xxx01101	xx01101x
n+14		DSCE	xxx01110	xx01110x
n+15		DSCF	xxx01111	xx01111x

x = don't care
*n is the value in the IRP x 16
**Master CPU accesses the register address in Register 5.

Table 3. Master CPU/UPC Register Map

Special Configurations

The Protopack version of the UPC is identical to the 40-pin ROM-based UPC with the following exceptions:

- All but 36 bytes of internal ROM are omitted from the Protopack RAM/ROM version.
- The memory address and data lines are buffered and brought out to the socket on the Protopack. This socket uses a 2Kx8 RAM or ROM.

The Protopack version of the UPC allows the user to prototype the system in hardware with an actual UPC device and to develop the code intended to be mask programmed into the on-chip ROM of the 40-pin UPC for the production system. The Protopack version of the UPC is an extremely versatile part. RAM program memory can be used on the 40-pin Protopack with RAM/ROM for all but 36 bytes of the UPC's memory space. This memory can then be downloaded from the master CPU using a bootstrap program stored in the 36 bytes (C-2F). Figure 10 is a memory map for the RAM version. This package will also accept a ROM, provided that the area from C to 2F is not used for programming.

Using the Z8094/Z8594 with EPROM or RAM. The Z8094 Z-UPC and the Z8594 UPC can be used with an EPROM or RAM plugged into the socket on top of the 40-pin package. Instructions for using a RAM are provided in Chapter 8 of the UPC Technical Manual (document #00-2055-01). If an EPROM is used, the following design considerations must be observed for proper operation:

1. The pin-out for the EPROM is 2716 compatible.
2. Programs in the EPROM must begin at 30 (hex). The RAM bootstrap ROM resides in locations 0C (hex) to 2F (hex).
3. The LDE instructions that would attempt a write to the EPROM *cannot* be used.
4. The UPC must be taken out of the Download mode by the host CPU after a reset. This is accomplished by having the host CPU write two bytes to the UPC. The first byte must reset the Interrupt Pending bit (D5) in the Master CPU Interrupt Control (MIC) register. The second byte must set the End of Message bit (D0) in the same register.

Any static RAM that can be interchangeably used with a 2716 EPROM can be plugged into the Protopack socket.

Protopack Pin Functions. Forty of the pins on the Protopack versions have functions identical to those of the 40-pin version. The remaining 24 pins have additional functions described below. (Figure 11 shows the Protopack versions' pin functions and pin assignments.)

A_0-A_{10}. *Program Memory Address Lines* (output). These lines are identical in all RAM/ROM versions in the Protopack. They are used to address 2K bytes of external UPC memory.

D_0-D_7. *Program Data* (input/output). Data is read in from the external memory on these lines. The RAM version also writes external memory through this bus.

$\overline{MDS}$. *Memory Data Strobe* (output, active Low). This signal is Low during an instruction fetch or memory write.

MR/$\overline{W}$. *Memory Read/Write* (output RAM versions only). This signal is High when the UPC is fetching an instruction and Low when it is loading external memory.

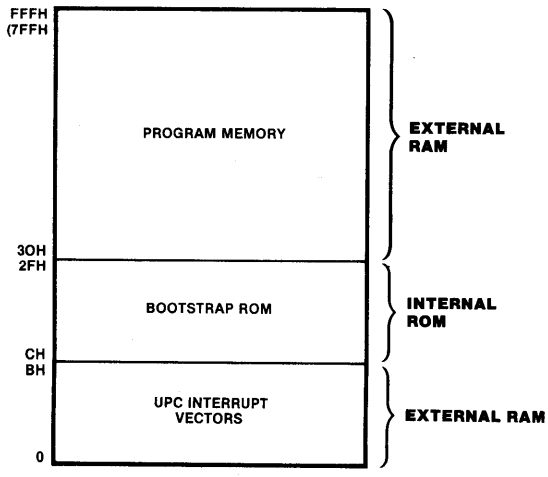


Figure 10. UPC RAM Version Memory Map

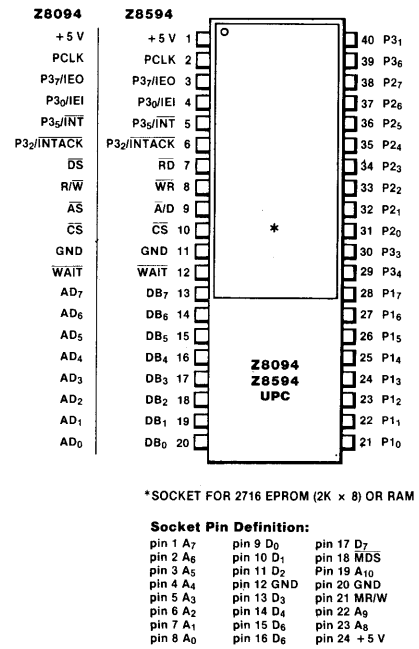


Figure 11. Z8094/Z8594 UPC Protopack Pin Assignments

Addressing Modes

The following notation is used to describe the addressing modes and instruction operations as shown in the instruction summary.

IRR	Indirect register pair or indirect working-register pair address
Irr	Indirect working-register pair only
X	Indexed address
DA	Direct address
RA	Relative address
IM	Immediate
R	Register or working-register address
r	Working-register address only
IR	Indirect-register or indirect working-register address
Ir	Indirect working-register address only
RR	Register pair or working-register pair address

Additional Symbols

dst	Destination location or contents
src	Source location or contents
cc	Condition code (see list)
@	Indirect address prefix
SP	Stack Pointer (control register FFH)
PC	Program Counter
FLAGS	Flag register (control register FCH)
RP	Register Pointer (control register FDH)
IMR	Interrupt Mask register (control register FBH)

Assignment of a value is indicated by the symbol "←". For example,

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

dst (7)

refers to bit 7 of the destination operand.

Flags

Control Register FCH contains the following six flags:

C	Carry flag
Z	Zero flag
S	Sign flag
V	Overflow flag
D	Decimal-adjust flag
H	Half-carry flag

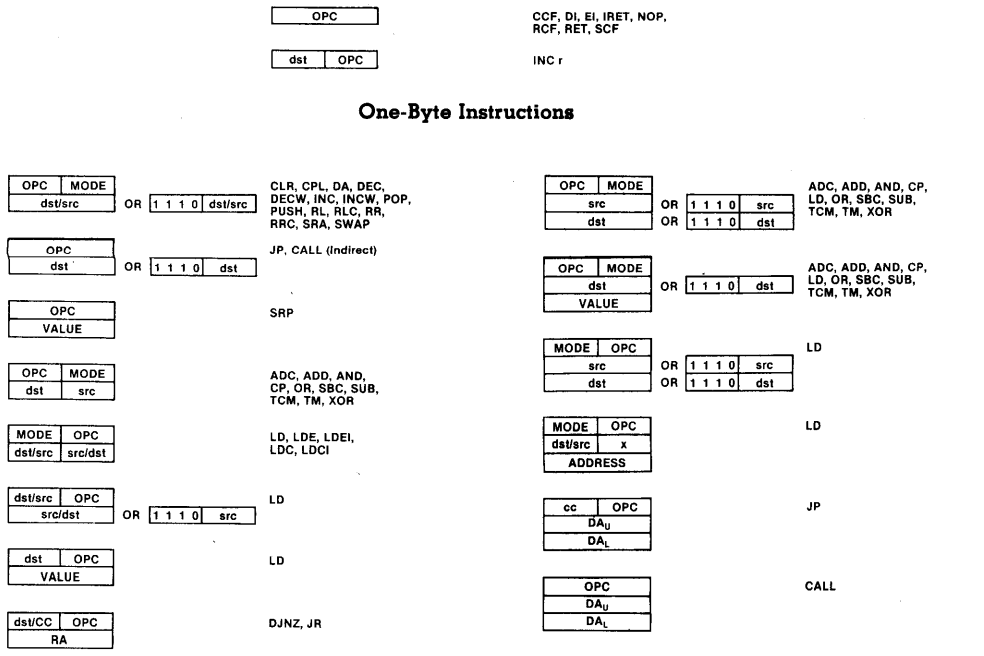
Affected flags are indicated by:

0	Cleared to zero
1	Set to one
*****	Set or cleared according to operation
–	Unaffected
X	Undefined

Condition Codes

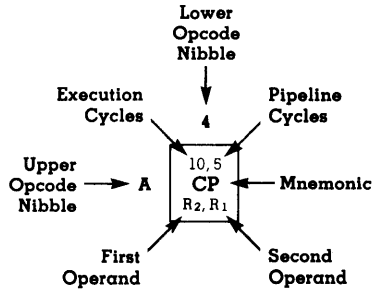
Value	Mnemonic	Meaning	Flags Set
1000		Always true	—
0111	C	Carry	C = 1
1111	NC	No carry	C = 0
0110	Z	Zero	Z = 1
1110	NZ	Not zero	Z = 0
1101	PL	Plus	S = 0
0101	MI	Minus	S = 1
0100	OV	Overflow	V = 1
1100	NOV	No overflow	V = 0
0110	EQ	Equal	Z = 1
1110	NE	Not equal	Z = 0
1001	GE	Greater than or equal	(S XOR V) = 0
0001	LT	Less than	(S XOR V) = 1
1010	GT	Greater than	[Z OR (S XOR V)] = 0
0010	LE	Less than or equal	[Z OR (S XOR V)] = 1
1111	UGE	Unsigned greater than or equal	C = 0
0111	ULT	Unsigned less than	C = 1
1011	UGT	Unsigned greater than	(C = 0 AND Z = 0) = 1
0011	ULE	Unsigned less than or equal	(C OR Z) = 1
0000		Never true	—

Instruction Formats



Opcode Map

Upper Nibble (Hex)	Lower Nibble (Hex) 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	6,5 **DEC** R_1	6,5 **DEC** IR_1	6,5 **ADD** r_1, r_2	6,5 **ADD** r_1, Ir_2	10,5 **ADD** R_2, R_1	10,5 **ADD** IR_2, R_1	10,5 **ADD** R_1, IM	10,5 **ADD** IR_1, IM	6,5 **LD** r_1, R_2	6,5 **LD** r_2, R_1	12/10,5 **DJNZ** r_1, RA	12/10,0 **JR** cc, RA	6,5 **LD** r_1, IM	12/10,0 **JP** cc, DA	6,5 **INC** r_1	
1	6,5 **RLC** R_1	6,5 **RLC** IR_1	6,5 **ADC** r_1, r_2	6,5 **ADC** r_1, Ir_2	10,5 **ADC** R_2, R_1	10,5 **ADC** IR_2, R_1	10,5 **ADC** R_1, IM	10,5 **ADC** IR_1, IM								
2	6,5 **INC** R_1	6,5 **INC** IR_1	6,5 **SUB** r_1, r_2	6,5 **SUB** r_1, Ir_2	10,5 **SUB** R_2, R_1	10,5 **SUB** IR_2, R_1	10,5 **SUB** R_1, IM	10,5 **SUB** IR_1, IM								
3	8,0 **JP** IRR_1	6,1 **SRP** IM	6,5 **SBC** r_1, r_2	6,5 **SBC** r_1, Ir_2	10,5 **SBC** R_2, R_1	10,5 **SBC** IR_2, R_1	10,5 **SBC** R_1, IM	10,5 **SBC** IR_1, IM								
4	8,5 **DA** R_1	8,5 **DA** IR_1	6,5 **OR** r_1, r_2	6,5 **OR** r_1, Ir_2	10,5 **OR** R_2, R_1	10,5 **OR** IR_2, R_1	10,5 **OR** R_1, IM	10,5 **OR** IR_1, IM								
5	10,5 **POP** R_1	10,5 **POP** IR_1	6,5 **AND** r_1, r_2	6,5 **AND** r_1, Ir_2	10,5 **AND** R_2, R_1	10,5 **AND** IR_2, R_1	10,5 **AND** R_1, IM	10,5 **AND** IR_1, IM								
6	6,5 **COM** R_1	6,5 **COM** IR_1	6,5 **TCM** r_1, r_2	6,5 **TCM** r_1, Ir_2	10,5 **TCM** R_2, R_1	10,5 **TCM** IR_2, R_1	10,5 **TCM** R_1, IM	10,5 **TCM** IR_1, IM								
7	10/12,1 **PUSH** R_2	12/14,1 **PUSH** IR_2	6,5 **TM** r_1, r_2	6,5 **TM** r_1, Ir_2	10,5 **TM** R_2, R_1	10,5 **TM** IR_2, R_1	10,5 **TM** R_1, IM	10,5 **TM** IR_1, IM								
8	10,5 **DECW** RR_1	10,5 **DECW** IR_1	12,0 **LDE** r_1, Irr_2	18,0 **LDEI** Ir_1, Irr_2												6,1 **DI**
9	6,5 **RL** R_1	6,5 **RL** IR_1	12,0 **LDE** r_2, Irr_1	18,0 **LDEI** Ir_2, Irr_1												6,1 **EI**
A	10,5 **INCW** RR_1	10,5 **INCW** IR_1	6,5 **CP** r_1, r_2	6,5 **CP** r_1, Ir_2	10,5 **CP** R_2, R_1	10,5 **CP** IR_2, R_1	10,5 **CP** R_1, IM	10,5 **CP** IR_1, IM								14,0 **RET**
B	6,5 **CLR** R_1	6,5 **CLR** IR_1	6,5 **XOR** r_1, r_2	6,5 **XOR** r_1, Ir_2	10,5 **XOR** R_2, R_1	10,5 **XOR** IR_2, R_1	10,5 **XOR** R_1, IM	10,5 **XOR** IR_1, IM								16,0 **IRET**
C	6,5 **RRC** R_1	6,5 **RRC** IR_1	12,0 **LDC** r_1, Irr_2	18,0 **LDCI** Ir_1, Irr_2				10,5 **LD** r_1, x, R_2								6,5 **RCF**
D	6,5 **SRA** R_1	6,5 **SRA** IR_1	12,0 **LDC** r_2, Irr_1	18,0 **LDCI** Ir_2, Irr_1	20,0 **CALL*** IRR_1		20,0 **CALL** DA	10,5 **LD** r_2, x, R_1								6,5 **SCF**
E	6,5 **RR** R_1	6,5 **RR** IR_1		6,5 **LD** r_1, Ir_2	10,5 **LD** R_2, R_1	10,5 **LD** IR_2, R_1	10,5 **LD** R_1, IM	10,5 **LD** IR_1, IM								6,5 **CCF**
F	8,5 **SWAP** R_1	8,5 **SWAP** IR_1		6,5 **LD** Ir_1, r_2		10,5 **LD** R_2, IR_1										6,0 **NOP**
Bytes per Instruction	2				3				2					3	1	



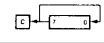
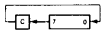
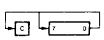
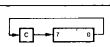
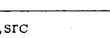
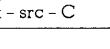
Legend:
R = 8-Bit Address
r = 4-Bit Address
R_1 or r_1 = Dst Address
R_2 or r_2 = Src Address

Sequence:
Opcode, First Operand, Second Operand

Note: The blank areas are not defined.

*2-byte instruction; fetch cycle appears as a 3-byte instruction.

Instruction Summary

Instruction and Operation	Addr Mode dst	Addr Mode src	Opcode Byte (Hex)	Flags Affected C Z S V D H
ADC dst,src dst ← dst + src + C	(Note 1)		1□	* * * * 0 *
ADD dst,src dst ← dst + src	(Note 1)		0□	* * * * 0 *
AND dst,src dst ← dst AND src	(Note 1)		5□	- * * 0 - -
CALL dst SP ← SP - 2 @SP ← PC; PC ← dst	DA IRR		D6 D4	- - - - - -
CCF C ← NOT C			EF	* - - - - -
CLR dst dst ← 0	R IR		B0 B1	- - - - - -
COM dst dst ← NOT dst	R IR		60 61	- * * 0 - -
CP dst,src dst - src	(Note 1)		A□	* * * * - -
DA dst dst ← DA dst	R IR		40 41	* * * X - -
DEC dst dst ← dst - 1	R IR		00 01	- * * * - -
DECW dst dst ← dst - 1	RR IR		80 81	- * * * - -
DI IMR (7) ← 0			8F	- - - - - -
DJNZ r,dst r ← r - 1 if r ≠ 0 PC ← PC + dst Range: +127, -128	RA		rA r = 0-F	- - - - - -
EI IMR (7) ← 1			9F	- - - - - -
INC dst dst ← dst + 1	r R IR		rE r = 0-F 20 21	- * * * - -
INCW dst dst ← dst + 1	RR IR		A0 A1	- * * * - -
IRET FLAGS ← @SP; SP ← SP + 1 PC ← @SP; SP ← SP + 2; IMR (7) ← 1			BF	* * * * * *
JP cc,dst if cc is true PC ← dst	DA IRR		cD c = 0-F 30	- - - - - -
JR cc,dst if cc is true, PC ← PC + dst Range: +127, -128	RA		cB c = 0-F	- - - - - -
LD dst,src dst ← src	r r R r X r Ir R R R IR IM	IM R r X r Ir r R IR IM IM R	rC r8 r9 r = 0-F C7 D7 E3 F3 E4 E5 E6 E7 F5	- - - - - -
LDC dst,src dst ← src	r Irr	Irr r	C2 D2	- - - - - -
LDCI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	C3 D3	- - - - - -
LDE dst,src dst ← src	r Irr	Irr r	82 92	- - - - - -
LDEI dst,src dst ← src r ← r + 1; rr ← rr + 1	Ir Irr	Irr Ir	83 93	- - - - - -
NOP			FF	- - - - - -
OR dst,src dst ← dst OR src	(Note 1)		4□	- * * 0 - -
POP dst dst ← @SP SP ← SP + 1	R IR		50 51	- - - - - -
PUSH src SP ← SP - 1; @SP ← src		R IR	70 71	- - - - - -
RCF C ← 0			CF	0 - - - - -
RET PC ← @SP; SP ← SP + 2			AF	- - - - - -
RL dst 	R IR		90 91	* * * * - -
RLC dst 	R IR		10 11	* * * * - -
RR dst 	R IR		E0 E1	* * * * - -
RRC dst 	R IR		C0 C1	* * * * - -
SBC dst,src dst ← dst - src - C	(Note 1)		3□	* * * * 1 *
SCF C ← 1			DF	1 - - - - -
SRA dst 	R IR		D0 D1	* * * 0 - -
SRP src RP ← src		Im	31	- - - - - -
SUB dst,src dst ← dst - src	(Note 1)		2□	* * * * 1 *
SWAP dst 	R IR		F0 F1	X * * X - -
TCM dst,src (NOT dst) AND src	(Note 1)		6□	- * * 0 - -
TM dst,src dst AND src	(Note 1)		7□	- * * 0 - -
XOR dst,src dst ← dst XOR src	(Note 1)		B□	- * * 0 - -

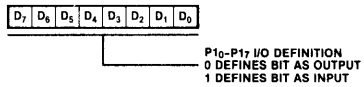
Note 1

These instructions have an identical set of addressing modes, which are encoded for brevity. The first opcode nibble is found in the instruction set table above. The second nibble is expressed symbolically by a □ in this table, and its value is found in the following table to the left of the applicable addressing mode pair.

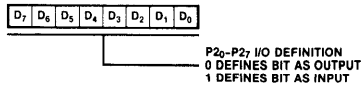
For example, to determine the opcode of an ADC instruction using the addressing modes r (destination) and Ir (source) is 13.

Addr Mode dst	Addr Mode src	Lower Opcode Nibble
r	r	2
r	Ir	3
R	R	4
R	IR	5
R	IM	6
IR	IM	7

R248 P1M
Port 1 Mode Register
UPC register address (Hex): F8



R246 P2M
Port 2 Mode Register
UPC register address (Hex): F6



R247 P3M
Port 3 Mode Register
UPC register address (Hex): F7

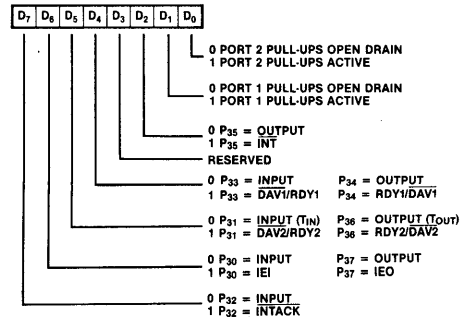
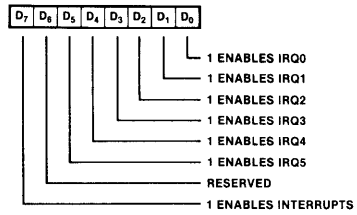
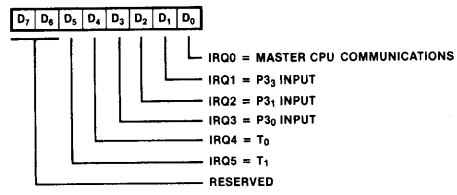


Figure 12. Port Mode Registers

R251 IMR
Interrupt Mask Register
UPC register address (Hex): FB



R250 IRQ
Interrupt Request Register
UPC register address (Hex): FA



R249 IPR
Interrupt Priority Register
UPC register address (Hex): F9 (Write Only)

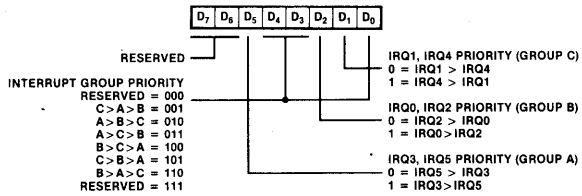
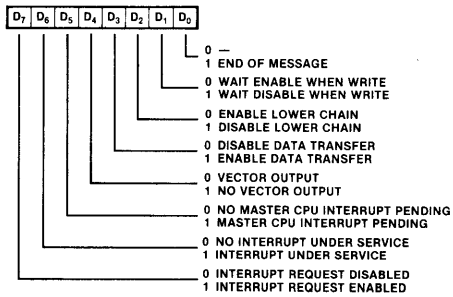


Figure 13. Interrupt Control Registers

R254 MIC
Master CPU Interrupt Control Register
UPC register address (Hex): FE



R240 MIV
Master CPU Interrupt Vector Register
UPC register address (Hex): F0

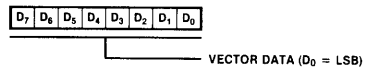
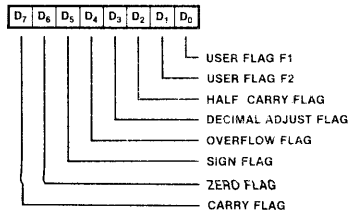


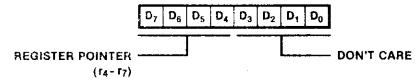
Figure 14. Master CPU Interrupt Registers

Registers
(Continued)

R252 FLAGS
Flag Register
UPC register address (Hex): FC



R253 RP
Register Pointer
UPC register address (Hex): FD



R255 SP
Stack Pointer
UPC register address (Hex): FF

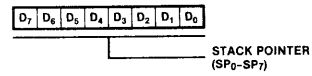
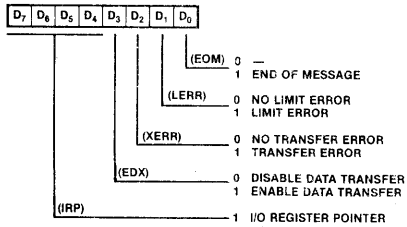
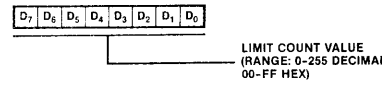


Figure 15. UPC Control Registers

R0 DTC
Data Transfer Control Register
UPC register address (Hex): 00



R4 LC
Limit Count Register
UPC register address (Hex): 04



R5 DIND
Data Indirection Register
UPC register address (Hex): 05

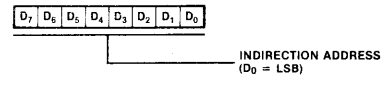
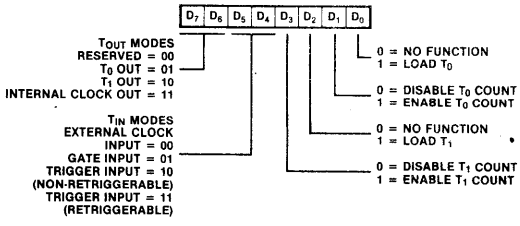
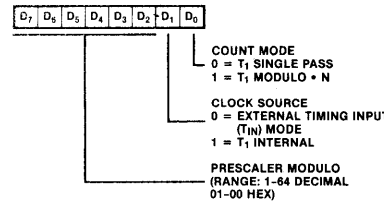


Figure 16. Master CPU-UPC Data Transfer Registers

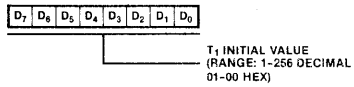
R241 TMR
Timer Mode Register
UPC register address (Hex): F1



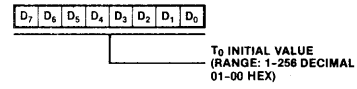
R243 PRE1
Prescaler 1 Register
UPC register address (Hex): F3



R242 T1
Counter/Timer 1 Register
UPC register address (Hex): F2



R244 T0
Counter/Timer 0 Register
UPC register address (Hex): F4



R245 PRE0
Prescaler 0 Register
UPC register address (Hex): F5

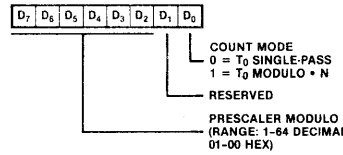


Figure 17. UPC Counter/Timer Registers

Registers (Continued)

Control Register	D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Comments
00_H Data Transfer Control Register	X	X	X	X	0	0	0	0	Disable data transfer from master CPU
04_H Limit Count Register	Not Defined								
05_H Data Indirection Register	Not Defined								
$F0_H$ Interrupt Vector Register	Not Defined								
$F1_H$ Timer Mode	0	0	0	0	0	0	0	0	Stops T0 and T1
$F2_H$ T0 Register	Not Defined								
$F3_H$ T0 Prescaler	X	X	X	X	X	X	0	0	Single-Pass mode
$F4_H$ T1 Register	Not Defined								
$F5_H$ T1 Prescaler	X	X	X	X	X	X	0	0	Single-Pass mode External clock source
$F6_H$ Port 2 Mode	1	1	1	1	1	1	1	1	Port 2 lines defined as inputs
$F7_H$ Port 3 Mode	0	0	0	0	X	1	0	0	Port 1, 2 open drain; $P3_5$ = INT; $P3_0$, $P3_1$, $P3_2$, $P3_3$ defined as input; $P3_4$, $P3_6$, $P3_7$ defined as output.
$F8_H$ Port 1 Mode	1	1	1	1	1	1	1	1	Port 1 lines defined as inputs
$F9_H$ Interrupt Priority	Not Defined								
FA_H Interrupt Request	X	X	0	0	0	0	0	0	Reset Interrupt Request
FB_H Interrupt Mask	0	X	X	X	X	X	X	X	Interrupts disabled
FC_H Flag Register	Not Defined								
FD_H Register Pointer	Not Defined								
FE_H Master CPU Interrupt Control Register	0	0	0	0	0	0	0	0	Master CPU interrupt disabled; wait enable when write; lower chain enabled
FF_H Stack Pointer	Not Defined								

NOTE: X means not defined.

Table 4. Control Register Reset Conditions

Absolute Maximum Ratings

Voltages on all pins with respect to GND . . . . . -0.5 V to +7.0 V

Operating Ambient Temperature . . . . . See Ordering Information

Storage Temperature . . . . . -65°C to +150°C

Stresses greater than those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; operation of the device at any condition above those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Standard Test Conditions

The characteristics below apply for the following standard test conditions, unless otherwise noted. All voltages are referenced to GND. Positive current flows into the reference pin. Standard conditions are as follows:

- 4.75 V ≤ V_{CC} ≤ +5.25 V
- V_{SS} = GND = 0 V
- 0°C ≤ T_A ≤ +70°C*

*See Ordering Information section for package temperature range and product number.

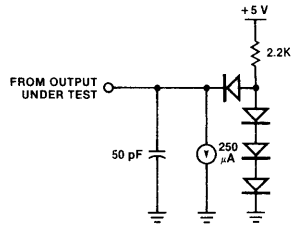


Figure 18. Test Load 1

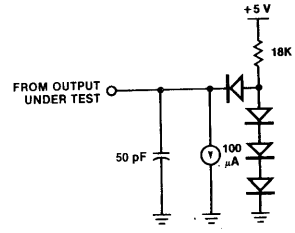


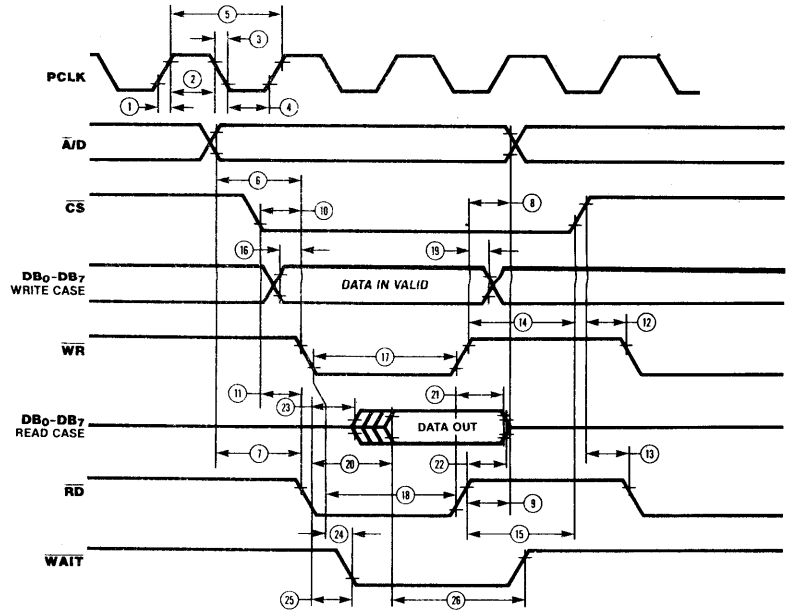
Figure 19. Test Load 2

DC Characteristics

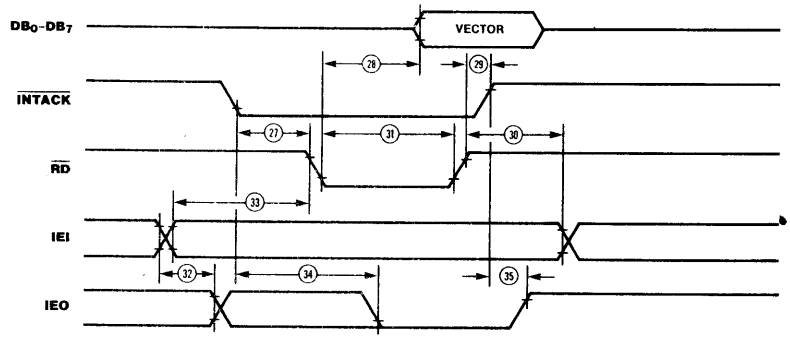
Symbol	Parameter	Min	Max	Unit	Condition	Notes
V_{CH}	Clock Input High Voltage	2.4	V_{CC}	V		
V_{CL}	Clock Input Low Voltage	-0.3	0.8	V		
V_{IH}	Input High Voltage	2.0	V_{CC}	V		
V_{IL}	Input Low Voltage	-0.3	0.8	V		
V_{OH}	Output High Voltage	2.4		V	I_{OH} = -250 µA	1
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = +2.0 mA	1
I_{IL}	Input Leakage	-10	10	µA	0 ≤ V_{IN} ≤ +5.25 V	
I_{OL}	Output Leakage	-10	10	µA	0 ≤ V_{IN} ≤ +5.25 V	
I_{CC}	V_{CC} Supply Current		180	mA		2

1. For A_0-A_{11}, D_0-D_7, and MR/$\overline{W}$ on the Protopack versions, I_{OH} = 100 µA and I_{OL} = 1.0 mA.
2. For Protopack versions, I_{CC} = 180 mA plus the current for the memory IC used.

Master CPU Interface Timing Z8590/94



Interrupt Acknowledge Timing Z8590/94

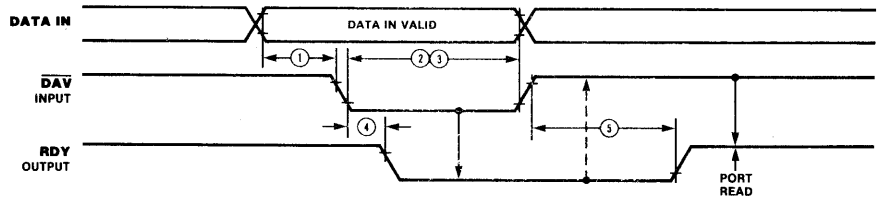


No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz** Min	6 MHz** Max	Notes*†
1	TrC	Clock Rise Time		20		15	
2	TwCH	Clock High Width	105	1855	70	1855	
3	TfC	Clock Fall Time		20		10	
4	TwCl	Clock Low Width	105	1855	70	1855	
5	TpC	Clock Period	250	2000	165	2000	
6	TsA/D(WR)	A/D to WR ↓ Setup Time	80		80		
7	TsA/D(RD)	A/D to RD ↓ Setup Time	80		80		
8	ThA/D(WR)	A/D to WR ↑ Hold Time	30		25		
9	ThA/D(RD)	A/D to RD ↑ Hold Time	30		25		
10	TsCSf(WR)	CS ↓ to WR ↓ Setup Time	0		0		
11	TsCSf(RD)	CS ↓ to RD ↓ Setup Time	0		0		
12	TsCSr(WR)	CS ↑ to WR ↓ Setup Time	60		60		
13	TsCSr(RD)	CS ↑ to RD ↓ Setup Time	60		60		
14	ThCS(WR)	CS to WR ↓ Hold Time	0		0		
15	ThCS(RD)	CS to RD ↓ Hold Time	0		0		
16	TsDI(WR)	Data in to WR ↓ Setup Time	0		0		
17	Tw(WR)	WR Low Width	390		250		
18	Tw(RD)	RD Low Width	390		250		
19	ThWR(DI)	Data in to WR ↑ Hold Time	0		0		
20	TdRD(DI)	Data Valid from RD ↓ Delay					1
21	ThRD(DI)	Data Valid to RD ↑ Hold Time	0		0		
22	TdRD(DI_Z)	Data Bus Float Delay from RD ↑		70		45	
23	TdRD(DB_A)	RD ↓ to Read Data Active Delay	0		0		
24	TdWR(W)	WR ↓ to WAIT ↓ Delay		150		150	
25	TdRD(W)	RD ↓ to WAIT ↓ Delay		150		150	
26	TdDI(W)	Data Valid to WAIT ↑ Delay	0		0		
27	TsACK(RD)	INTACK ↓ to RD ↓ Setup Time	90		80		2
28	TdRD(DI)	RD ↓ to Vector Valid Delay		255		180	
29	ThRD(ACK)	RD ↑ to INTACK ↑ Hold Time	0		0		
30	ThIEI(RD)	IEI to RD ↑ Hold Time	100		100		
31	TwRDl	RD (Acknowledge) Low Width	255		250		
32	TdIEI(IEO)	IEI to IEO Delay		120		100	
33	TsIEI(RD)	IEI to RD ↓ Setup Time	150		120		
34	$TdACK_f$(IEO)	INTACK ↓ to IEO ↓ Delay		250		250	
35	$TdADK_r$(IEO)	INTACK ↑ to IEO ↑ Delay		250		250	

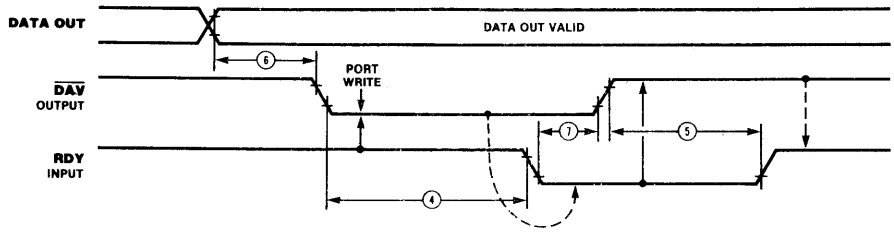
NOTES:
1. This parameter is dependent on the state of the UPC at the time of master CPU access.
2. In case where daisy chain is not used.
3. The timing characteristics given reference 2.0 V as High and 0.8 V as Low.
4. All output ac parameters use test load 1.

*Timings are preliminary and subject to change.
†Units in nanoseconds (ns).
**Z8590 only.

Handshake Timing

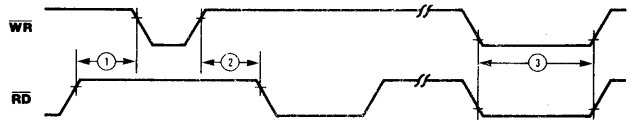


Input Handshake

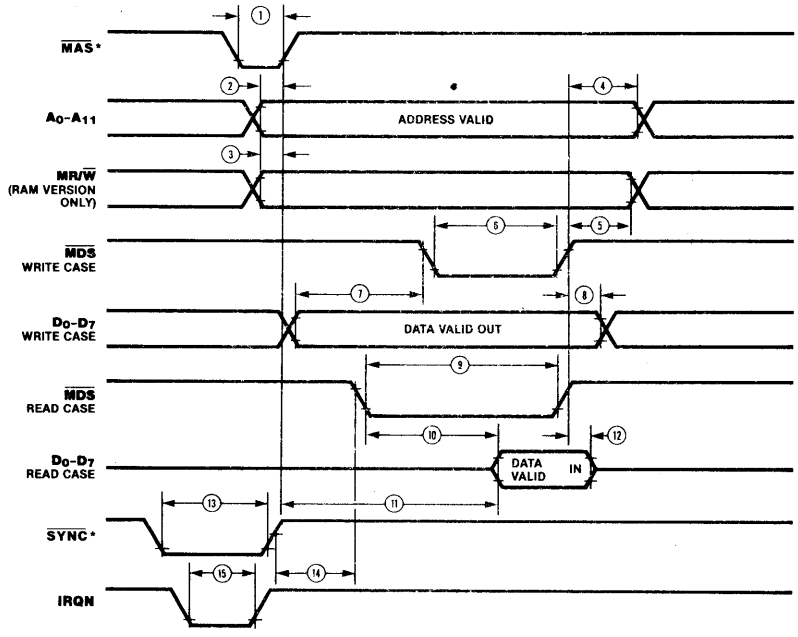


Output Handshake

Reset Timing



RAM Version Program Memory Timing



*This signal is not available externally.

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz** Min	6 MHz** Max	Notes*†
1	TsDI(DA)	Data in Setup Time	0		0		
2	ThDA(DI)	Data in Hold Time	230		230		
3	TwDA	Data Available Width	175		175		1,2
4	TdDAL(RY)	Data Available Low to Ready Delay Time	20 0	175	20 0	175	1,2 2,3
5	TdDAH(RY)	Data Available High to Ready Delay Time	0	150	0	150	1,2 2,3
6	TdDO(DA)	Data Out to Data Available Delay Time	50		50		2
7	TdRY(DA)	Ready to Data Available Delay Time	0	205	0	205	2

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz** Min	6 MHz** Max	Notes*†
1	TdRDQ(WR)	Delay from $\overline{RD}$ ↑ to $\overline{WR}$ ↓ for No Reset	40		35		
2	TdWRQ(RD)	Delay from $\overline{WR}$ ↑ to $\overline{RD}$ ↓ for No Reset	50		35		
3	TwRES	Minimum Width of $\overline{WR}$ and $\overline{RD}$ both Low for Reset	250		250		4

No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz** Min	6 MHz** Max	Notes*†
1	TwMAS	Memory Address Strobe Width	60		55		5
2	TdA(MAS)	Address Valid to Memory Address Strobe ↑ Delay	30		30		5
3	TdMR/W(MAS)	Memory Read/Write to Memory Address Strobe ↑ Delay	30		30		5
4	TdMDS(A)	Memory Data Strobe ↑ to Address Change Delay	60		60		
5	TdMDS(MR/W)	Memory Data Strobe ↑ to Memory Read/Write Not Valid Delay	80		75		
6	Tw(MDS)	Memory Data Strobe Width (Write Case)	160		110		6
7	TdDO(MDS)	Data Out Valid to Memory Data Strobe ↓ Delay	30		30		5
8	TdMDS(DO)	Memory Data Strobe ↑ to Data Out Change Delay	30		30		5
9	Tw(MDS)	Memory Data Strobe Width (Read Case)	230		230		6
10	TdMDS(DI)	Memory Data Strobe ↓ to Data In Valid Delay		160		130	7
11	TdMAS(DI)	Memory Address Strobe ↑ to Data In Valid Delay		280		220	7
12	ThMDS(DI)	Memory Data Strobe ↑ to Data In Hold Time	0		0		
13	TwSY	Instruction Sync Out Width	160		100		
14	TdSY(MDS)	Instruction Sync Out to Memory Data Strobe Delay	200		160		
15	TwI	Interrupt Request via Port 3 Input Width	100		100		

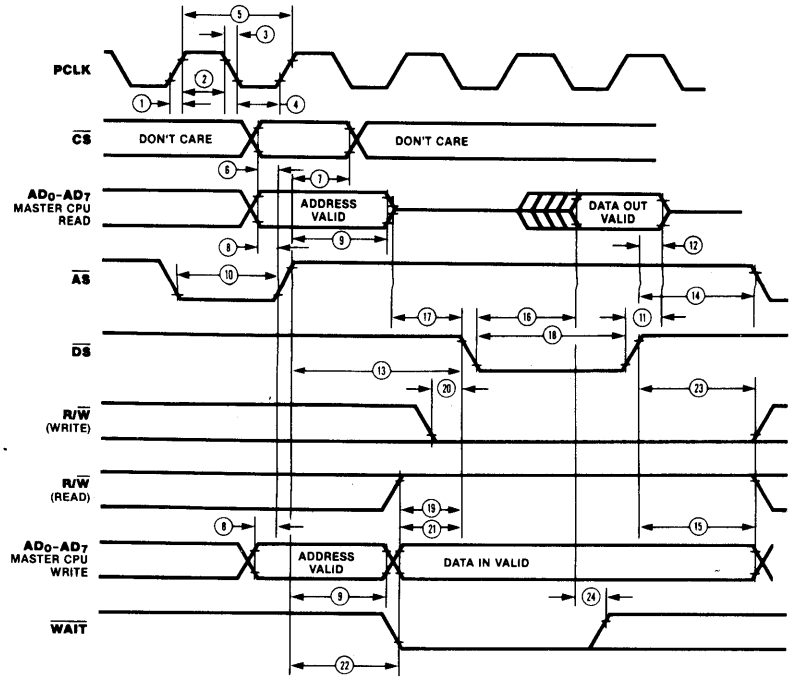
NOTES:
1. Input Handshake.
2. Test Load 1.
3. Output Handshake.
4. Internal reset signal is ½ to 2 clock delays from external reset condition.
5. Delay times are specified for an input clock frequency of 4 MHz. When operating at a lower frequency, the increase in input clock period must be added to the specified delay time.
6. Data strobe width is specified for an input clock frequency of 4 MHz. When operating at a lower frequency, the increase in three input clock periods must be added to the specified width. Data strobe width varies according to the instruction being executed.
7. Address strobe and data strobe to data in valid delay times represent memory system access times and are given for a 4 MHz input frequency.

*All timing references assume 2.0 V for a logic "1" and 0.8 V for a logic "0". All output ac parameters use test load 2. Timings are preliminary and subject to change.

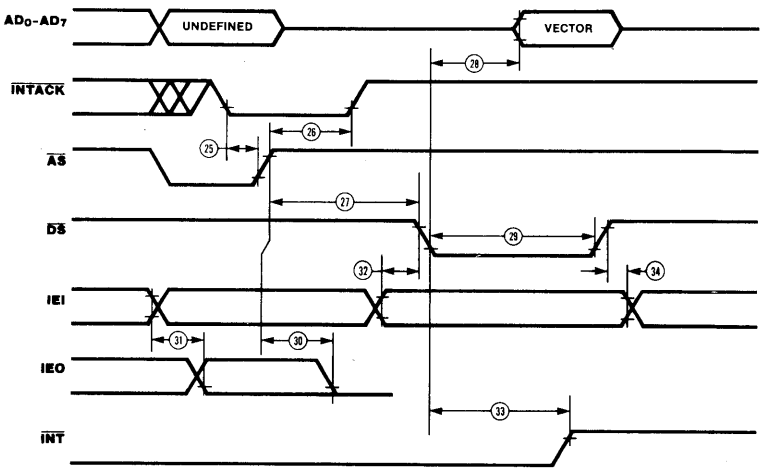
†Units in nanoseconds (ns).

**Z8090 only.

Master CPU Interface Timing Z8090/94



Interrupt Acknowledge Timing Z8090/94



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz** Min	6 MHz** Max	Notes*†
1	TrC	Clock Rise Time		20		15	
2	TwCh	Clock High Width	105	1855	70	1855	
3	TfC	Clock Fall Time		20		10	
4	TwCl	Clock Low Width	105	1855	70	1855	
5	TpC	Clock Period	250	2000	165	2000	
6	TsCS(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
7	ThCS(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
8	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
9	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
10	TwAS	$\overline{AS}$ Low Width	70		50		
11	TdDS(DR)	$\overline{DS}$ ↑ to Read Data Not Valid	0		0		
12	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70		45	2
13	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60	2095	40	2095	
14	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		35		
15	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		1
16	TdDS(DR)	$\overline{DS}$ ↓ to Read Data Valid Delay					3
17	TdAz(DS)	Address Float to $\overline{DS}$ Delay	0		0		
18	TwDS	$\overline{DS}$ Low Width	390		250		
19	TsRWR(DS)	R/$\overline{W}$ (Read) to $\overline{DS}$ ↓ Setup Time	100		80		
20	TsRWW(DS)	R/$\overline{W}$ (Write) to $\overline{DS}$ ↓ Setup Time	0		0		
21	TsDW(DSf)	Write Data to $\overline{DS}$ ↓ Setup Time	30		20		
22	TdAS(W)	$\overline{AS}$ ↑ to $\overline{WAIT}$ ↓ Valid Delay		195		160	
23	ThRW(DS)	R/$\overline{W}$ to $\overline{DS}$ ↑ Hold Time	60		40		
24	TsDR(W)	Read Data Valid to $\overline{WAIT}$ ↑	0		0		
25	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
26	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
27	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ (/Acknowledge) Delay	940		200		
28	TdDSA(DR)	$\overline{DS}$ ↓ (Acknowledge) to Read Data Valid Delay		360		180	
29	TwDSA	$\overline{DS}$ ↓ (Acknowledge) Low Width	475		250		
30	TdAS(IEO)	$\overline{AS}$ ↑ to IEO Delay		290		250	
31	TdIEIf(IEO)	IEI to IEO Delay		120		100	
32	TsIEI(DSA)	IEI to $\overline{DS}$ ↓ (Acknowledge) Setup Time	150		120		
33	TdDS(INT)	$\overline{DS}$ ↓ to $\overline{INT}$ Delay		500		500	
34	ThIEI(DS)	IEI to $\overline{DS}$ ↑ Hold Time	100		100		

NOTES:
1. Parameter does not apply to Interrupt Acknowledge transactions.
2. The maximum value for TdAS(DS) does not apply to Interrupt Acknowledge transactions.
3. This parameter is dependent on the state of UPC at the time of master CPU access.

* Timings are preliminary and subject change.
† Units in nanoseconds (ns).
**Z8090 only.
The timing characteristics given reference 2.0 V as High and 0.8 V as Low.
All output ac parameters use test load 1.

Product Number	Package/ Temp	Speed	Description	Product Number	Package/ Temp	Speed	Description
Z8590	CS	4.0 MHz	UPC (40-pin)	Z8090	PS	4.0 MHz	UPC (40-pin)
Z8090	CS	4.0 MHz	Same as above	Z8590-6	PS	6.0 MHz	Same as above
Z8590-6	CS	6.0 MHz	Same as above	Z8090-6	PS	6.0 MHz	Same as above
Z8090-6	CS	6.0 MHz	Same as above	Z8594	RS	4.0 MHz	UPC Protopack
Z8590	PS	4.0 MHz	Same as above	Z8094	RS	4.0 MHz	Same as above

NOTES: C = Ceramic, P = Plastic, R = Protopack; S = 0°C to +70°C.

Z800

Family

Zilog

Pioneering the Microworld

Zilog Z800™ Family

Beyond Compromise

September 1983

The advancing demands of the marketplace have caught many design projects between the need for new features and the desire to preserve existing software. The Z800 family provides the most desired new features without compromise. It is totally software- (object code) compatible with the Z80®.

Memory Management. The demand for memory expanded beyond 64K is satisfied by the memory management of the Z800, which also provides protection and dynamic relocation. Two alternatives, 512K bytes and 16M bytes of physical address space, are available with MPUs in 40- and 64-pin packages, respectively. The implementation deals with 4K or 8K pages, which can be individually mapped and protected. Virtual memory is also supported, via an instruction abort mechanism.

The utility of the larger spaces is enhanced by the optional separation of program and data and the implementation of System and User modes. System program, system data, user program, and user data can now each occupy 64K physical spaces without operating system intervention. Much larger spaces can be made available by reloading the on-chip control registers.

Performance. To fully benefit from longer programs operating on larger blocks of data, increased performance is essential. This is provided by the Z800 in several ways. First, a range of clock speeds will be available starting with 10 MHz. To achieve full benefit from the higher CPU speeds without making inordinate demands on memory access, on-chip cache memory featuring automatic update has been implemented.

Furthermore, a 16-bit bus option doubles the bandwidth to memory. For CPU-intensive applications, the system can be further tailored for cost-effective use of memory with the programmable bus-clock scaler and the four programmable wait-state generators. These devices operate directly off the on-chip clock oscillator, which runs at half the frequency of the crystal. Additional performance enhancement results from new addressing modes and new instructions. For example, hardware multiply and divide are standard in both signed and unsigned modes for both 8 and 16 bits.

Savings. System cost has been reduced through the integration of peripheral functions on-chip. Both board space and development time are saved by the use of the four DMAs, four counter/timers, and one UART available on the Z800 chip. These features—along with the refresh generator, memory manager, cache memory, and clock oscillator with the CPU—result in a virtual mainframe on-chip.

Support. MPU features can be converted to system benefits only with appropriate development suport tools. Zilog support starts with the UNIX* environment on the DEC-VAX or the Zilog System 8000. Cross-software is provided that includes an assembler, instruction-level simulator, C compiler, linker, loader, and library. This software is compatible with DEC-VMS and CP/M.

The Z-SCAN 800 is provided for debugging hardware and software. It features real-time emulation with menu-driven, screen-oriented software. Early system prototyping can be done with the single-board computer/development module (SBC/DM). This multibus board has 128K bytes of random access memory, four sockets for read-only memory, three serial ports, one parallel port, and an interface to floppy disks. Software for the SBC/DM will include CP/M and all the cross-software packages.

*UNIX is a trademark of Bell Laboratories.

Z800™ MPU Family

Zilog

Preliminary Product Specification

September 1983

FEATURES

- Enhanced Z80® instruction set that maintains object-code compatibility with Z80 microprocessor.
- On-chip paged Memory Management Unit (MMU).
- Large memory address space: 512K byte and 16M byte versions.
- On-chip, high-speed local or cache memory.
- High performance 16-bit Z-BUS interface or 8-bit Z80-compatible bus interface.
- Four on-chip 16-bit counter/timers.
- Four on-chip DMA channels.
- On-chip full duplex UART.
- 10-25 MHz CPU processor clock.

GENERAL DESCRIPTION

Zilog's new Z800 family of 8- and 16-bit microprocessors features high-performance microprocessors designed to give the end-user a powerful and cost effective solution to application requirements. The family consists of the 8-bit Z80-Bus microprocessors that are packaged in 40- and 64-pin dual in-line packages, and 16-bit Z-BUS microprocessors in 40- and 64-pin packages. The Z800 family incorporates advanced architectural features that allow fast and efficient throughput and increased memory addressing while maintaining Z80 object code compatibility. Z800 microprocessors offer both a continuing growth path for present Z80-based designs and a high-performance microprocessor for future designs.

Central to the Z800 microprocessors is an enhanced version of the Z80 Central Processing Unit (CPU). To assure system integrity, the Z800 microprocessors can operate in either user or system mode, allowing protection of system resources from user tasks and programs. System mode operation is supported by the addition of the system Stack Pointer to the working register set. The IX and IY registers have been modified so that in addition to their regular function as index registers, each register can be accessed as a 16-bit general purpose register or as two single-byte registers.

The Z80 CPU instruction set has been retained, meaning that the Z800 microprocessors are completely binary-code compatible with present Z80 code. The basic addressing modes of the Z80 microprocessor have been augmented with the addition of Indexed mode with full 16-bit displacement, Program Counter Relative with 16-bit displacement, Stack Pointer Relative with 16-bit displacement, and Base Index mode. The new addressing modes are incorporated into many of the old Z80 CPU instructions, resulting in greater flexibility and power. Some additions to the instruction set include 8-and 16-bit signed and unsigned multiply and divide, 8-and 16-bit sign extension, and a test and set instruction to support multiprocessing. The 16-bit instructions have been expanded to include 16-bit compare, memory increment, memory decrement, negate, add, and subtract, in addition to the previously mentioned multiply and divide.

A requirement of many of today's microprocessor-based system designs is to increase the memory address space beyond the 64K byte range of typical 8-bit microprocessors. The Z800 microprocessors have an on-chip Memory Management Unit (MMU) that enables the microprocessors to address either 512K bytes or 16M bytes, depending on the device package. In addition to enabling the address space to be expanded, the MMU performs other memory management functions previously handled by dedicated off-chip memory management devices.

I/O address space has been expanded by the addition of an I/O Page register used to select pages of I/O addresses. The 8-bit I/O Page register can select one of

256 possible pages of I/O addresses to be active at one time, allowing a total of 64K I/O addresses to be accessed.

There are 256 bytes of on-chip memory present on all members of the Z800 family. This memory can be configured as a high-speed cache or as a fixed address local memory. When configured as a cache, the memory can be programmed to be instruction only, data only, or both data and instruction. The cache memory allows programs to run significantly faster by reducing the number of external bus accesses. Operation and update of the cache is performed automatically and is completely transparent to the user. When used as a local memory, the addresses are programmable, allowing "RAMless" systems to be used.

Many features that have traditionally been handled by external peripheral devices have been incorporated in the design of the Z800 microprocessors. The "on-chip peripherals" reduce system chip count and reduce interconnection on the external bus. All members of the Z800 family contain an on-chip clock oscillator. Also present is a refresh controller that provides 10-bit refresh addresses for dynamic memories.

The 64-pin versions of the Z800 MPU contain additional on-chip peripherals to provide system design flexibility. To support high-bandwidth data transmission, four Direct Memory Access (DMA) channels are incorporated on-chip. Each DMA channel operates using full 24-bit source and destination addresses with a 16-bit count. The channels can be programmed to operate in single transaction, burst, or continuous mode. System event counting and timing requirements are met with the help of the four 16-bit counter/timers. The counter/timer functions can be externally controlled with gate and trigger inputs, and can be programmed as retriggerable or nonretriggerable. Also, a full duplex UART, capable of handling a variety of data and character formats, is present to facilitate asynchronous serial communication.

Regardless of whether the 8- or 16-bit bus is used, all members of the Z800 family feature programmable bus timing, allowing the user to tailor timing to the individual system. Upon reset the Z800 microprocessors can be programmed to have system timing that is one-fourth, one-half, or equal to the speed of the CPU, with one-half being the default. In addition to clock scaling, programmable wait states can be inserted during various bus transactions. Without the use of external hardware, one to three wait states can be inserted into memory, I/O, *and interrupt acknowledge transactions. Furthermore,* separate memory wait states can be specified for upper and lower memory areas, facilitating the use of different speeds of ROMs and RAMs in the same system.

An additional feature of the 16-bit bus interface is the ability to support "nibble-mode" dynamic RAMs. Using this feature (known as burst mode), the bus bandwidth of memory read transactions is essentially doubled. Burst mode transactions have the further benefit of allowing the cache to operate more efficiently by guaranteeing a high probability that the contents of the accessed memory will be present in the cache.

The Z800 family supports Zilog's Extended Processor Architecture (EPA) in a number of ways. All members are capable of trapping Extended Processor Unit (EPU) instructions in order to perform software emulation of the EPU. The Z8216 directly interfaces with an EPU such as the Z8070 Floating Point Unit and operates in a manner that is completely transparent to the user and the program. The other members of the Z800 family can interface easily with EPUs with the aid of support software.

The pin functions of four versions of the Z800 MPU, Z8108, Z8208, Z8116, and Z8216, are shown in Figures 1-4, respectively. A block diagram of the Z800 MPU is shown in Figure 5.

Z800 CPU

User and System Modes of Operation

The Z800 CPU can operate in either user or system mode. In user mode, some instructions cannot be executed and some registers of the CPU are inaccessible. In general, this mode of operation is intended for use by application programs. In system mode, all of the instructions can be executed and all of the CPU registers can be accessed. This mode is intended for use with programs that perform operating system functions. This separation of CPU resources promotes the integrity of the system, since programs operating in user mode cannot access those aspects of the CPU that deal with system interface events.

To further support the dual user/system mode, there are two copies of the Stack Pointer—one for the user stack and another for the system stack. These two stacks facilitate the task switching involved when interrupts or traps occur. To ensure that the user stack is free of system information, the information saved on the occurrence of interrupts or traps is always pushed onto the system stack before the new program status is loaded.

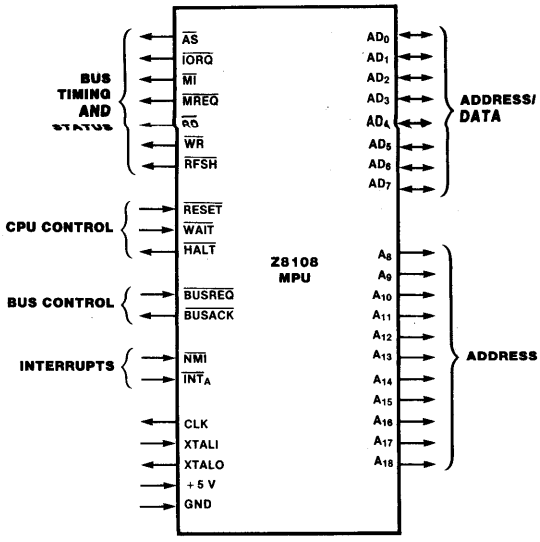


Figure 1. Z8108 Pin Functions

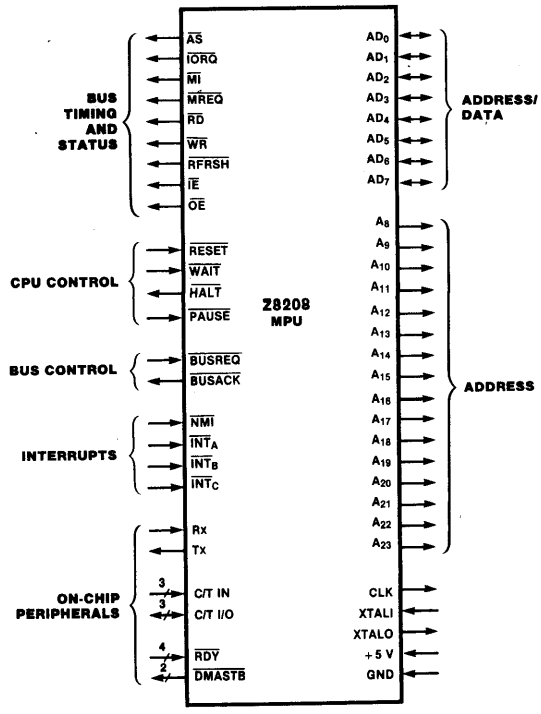


Figure 2. Z8208 Pin Functions

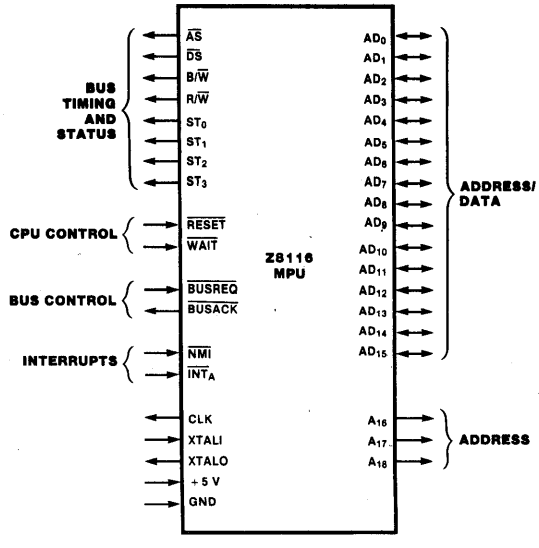


Figure 3. Z8116 Pin Functions

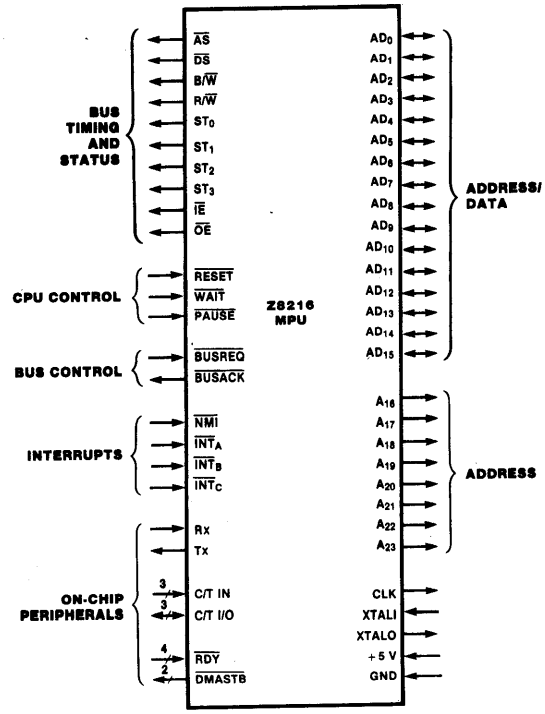


Figure 4. Z8216 Pin Functions

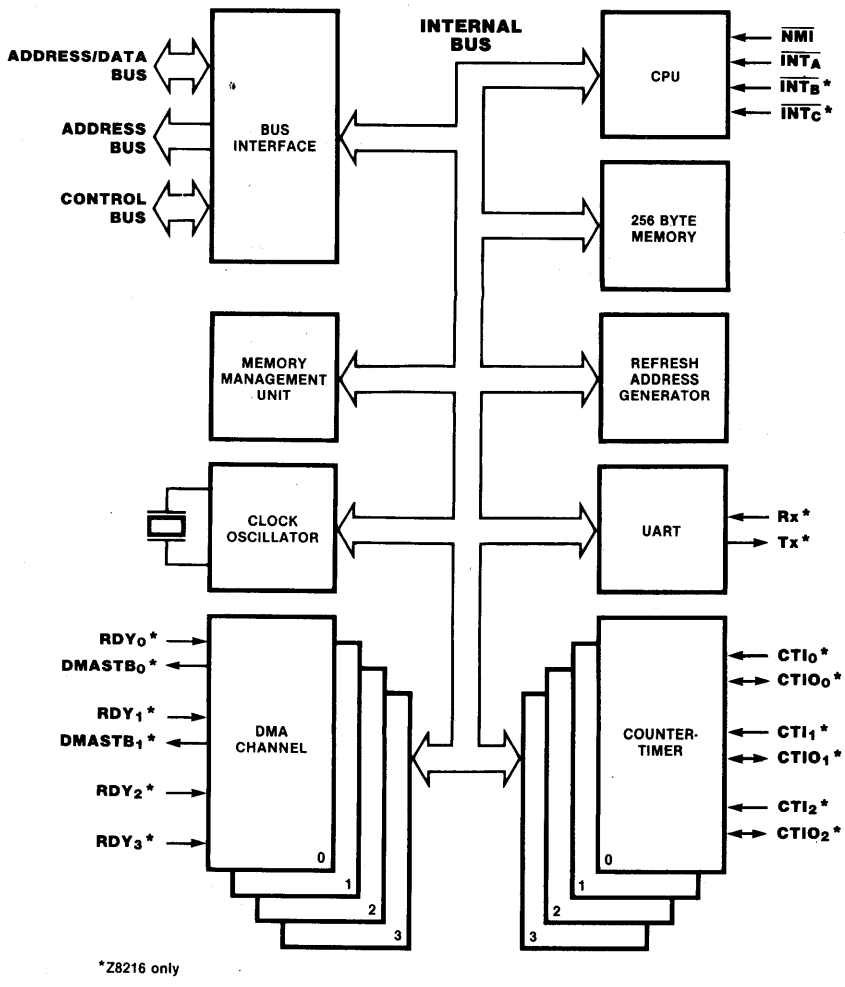


Figure 5. Z800 MPU Block Diagram

Address Spaces

The Z800 CPU architecture supports four distinct address spaces corresponding to the different types of locations that can be accessed by the CPU. These four address spaces are:

- CPU register space
- CPU control and status register space
- Memory address space
- I/O address space

CPU Register Space. The CPU register space consists of all of the registers in the CPU register file. The CPU registers are used for data and address manipulation. Access to these registers is specified in the instruction. The CPU registers are labeled F, A, B, C, D, E, H, L, F′, A′, B′, C′, D′, E′, H′, L′, IX, IY, SSP, USP, PC, I, and R.

CPU Control and Status Register Space. The CPU control register space consists of all of the control and status registers found in the CPU control register file. These registers govern the operation of the CPU and are accessible only by the privileged Load Control instruction. The registers in the CPU control file consist of the Master Status register, Bus Timing and Initialization register, Bus Timing and Control register, Interrupt/Trap Vector Table Pointer, I/O Page register, System Stack Limit register, Trap Control register, Interrupt Status register, Cache Control register, and Local Address register.

Memory Address Space. Two memory address spaces are supported by the Z800 CPU; one for user and one for system mode of operation. They are selected by the User/System Mode (U/S) bit in the Master Status register, which governs the selection of page descriptor registers during address translation.

Each address space can be viewed as a string of 64K bytes numbered consecutively in ascending order. The 8-bit byte is the basic addressable element in the memory address spaces. However, there are other addressable data elements: bits, 2-byte words, byte strings and multiple-byte EPU operands.

The address of a multiple-byte entity is the address of the byte with the lowest address. Multiple-byte entities can be stored beginning at either even or odd memory addresses.

I/O Address Space. I/O addresses are generated only by the I/O instructions IN, OUT, and the I/O block move instructions. Logical I/O addresses are eight bits in length, augmented by the A register on lines A_8–A_{15} in Direct Address addressing mode and by the B register on lines A_8–A_{15} in Indirect Register addressing mode and for block I/O instructions. The 16-bit logical I/O address is always extended by appending the contents of the 8-bit page register to the augmented I/O address. Thus the complete address generated to address an I/O port consists of an I/O page number on A_{23}–A_{16}, the contents of the A or B register on A_8–A_{15}, and the 8-bit I/O address on A_7–A_0.

Unlike memory references, in which a 16-bit word store or fetch can generate two memory references, an I/O word store or fetch is always one I/O bus transaction, regardless of bus size or I/O port address. Note, however, that on-chip peripherals with word registers are accessed via word I/O instructions for those 16-bit registers, regardless of the external bus size.

Data Types

The CPU can operate on bits, binary-coded decimal (BCD) digits (4 bits), bytes (8 bits), words (16 bits), byte strings, and word strings. Bits in registers or memory can be set, cleared, and tested. BCD digits, packed two to the byte, can be manipulated with the Decimal Adjust Accumulator instruction in conjunction with binary addition and subtraction. Bytes are operated on by 8-bit load, arithmetic, logical, and shift and rotate instructions. Words are operated on in a similar manner by the 16-bit load and 16-bit arithmetic instructions. Block move and search operations can manipulate byte strings up to 64K bytes long. Block I/O word instructions can manipulate word strings up to 32K words long. To support EPU operations, byte strings up to 16 bytes in length can be transferred by the CPU.

CPU Registers

The Z800 MPU contains 23 programmable registers in the CPU register address space. These registers are illustrated in Figure 6.

Primary and Working Register Set. The working register set is divided into the two 8-bit register files—the primary file and alternate (designated by ' [prime]) file. Each file contains an 8-bit accumulator (A), a Flag register (F), and six general-purpose registers (B, C, D, E, H, and L). Only one file can be active at any given time. Upon reset, the primary register file is active. Exchange instructions allow the programmer to exchange the active file with the inactive file.

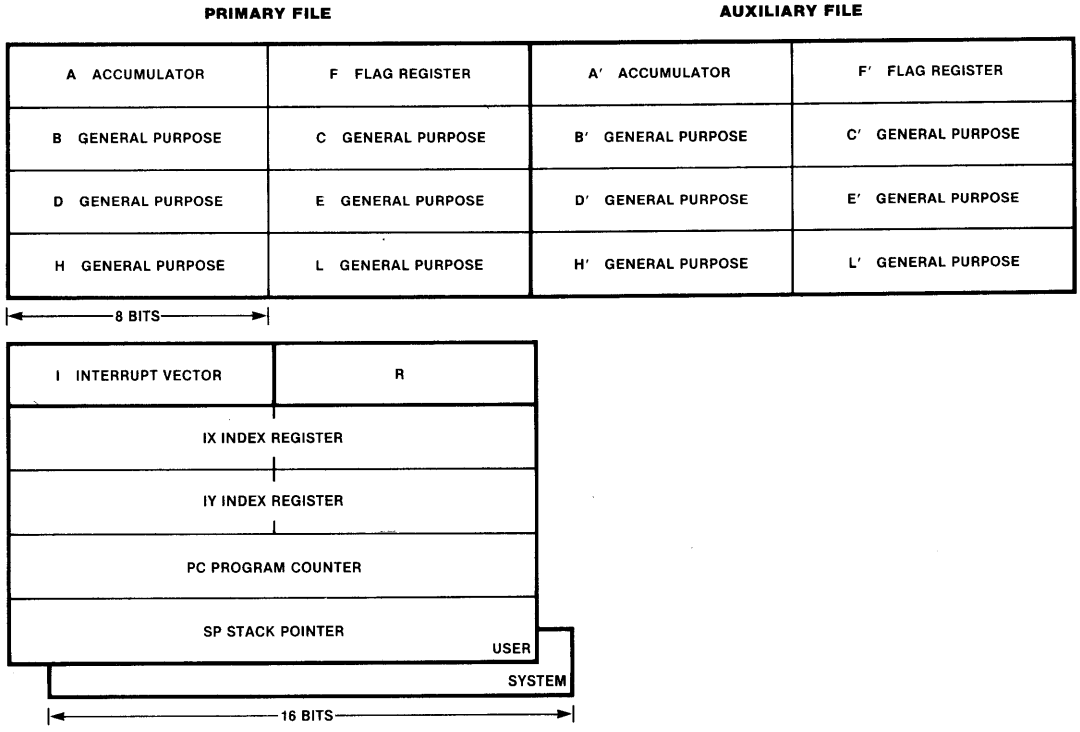


Figure 6. CPU Register Configuration

The accumulator is the destination register for 8-bit arithmetic and logical operations. The six general-purpose registers can be paired (BC, DE, and HL) to form three 16-bit general-purpose registers. The HL register pair serves as a 16-bit accumulator for 16-bit arithmetic operations.

CPU Flag Register. The Flag register contains six flags that are set or reset by various CPU operations. This register is illustrated in Figure 7.

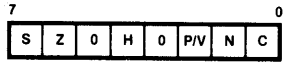


Figure 7. CPU Flag Register

The flags in this register are:

Carry (C). This flag is set when an add instruction generates a carry or a subtract instruction generates a borrow. Certain logical and rotate and shift instructions affect the Carry flag.

Add/Subtract (N). This flag is used by the Decimal Adjust Accumulator instruction to distinguish between add and subtract operations. The flag is set for subtract operations and cleared for addition operations.

Parity/Overflow (P/V). This flag is set or cleared depending on the operation being performed. During arithmetic operations it is set to indicate a twos complement overflow. During logical and rotate operations, this flag is set to indicate even parity of the result, or cleared to indicate odd parity.

Half Carry (H). This flag is set if an 8-bit arithmetic operation generates a carry or borrow between bits 3 and 4, or if a 16-bit operation generates a carry or borrow between bits 11 and 12. This bit is used to correct the result of a packed BCD addition or subtract operation.

Zero (Z). This flag is set if the result of an arithmetic or logical operation is a zero.

Sign (S). This flag stores the state of the most significant bit of the accumulator. The Sign flag is also used to indicate the results of a test and set instruction.

Dedicated CPU Registers

Index Registers. The two Index registers, IX and IY, each hold a 16-bit base address that is used in the Index addressing mode. The Index registers can also function as general-purpose registers with the upper and lower bytes capable of being accessed individually. The high and low bytes of the IX register are called IXH and IXL. The high and low bytes of the IY register are called IYH and IYL.

Interrupt Register. The Interrupt register (I) is used in interrupt mode 2 to generate a 16-bit indirect logical address to an interrupt service routine. The Interrupt register supplies the upper eight bits of the indirect address and the interrupting peripheral supplies the lower eight bits.

Program Counter. The Program Counter (PC) is used to sequence through instructions in the currently-executing program and to generate relative addresses. The Program Counter contains the 16-bit logical address of the current instruction being fetched from memory.

R Register. The R register can be used as a general-purpose 8-bit read/write register. The R register is not associated with the refresh address and its contents are changed only by the user.

Stack Pointers. Two hardware Stack Pointers, the user Stack Pointer (USP) and the system Stack Pointer (SSP), support the dual mode of operation of the microprocessor. The SSP is used for saving information when an interrupt or trap occurs, and for supporting subroutine calls and returns in system mode. The USP is used for supporting subroutine calls and returns in user mode.

Status and Control Registers. There are ten status and control registers available to the programmer in the Z800 MPU. Table 1 shows the addresses occupied by the registers in the status and control register addressing space.

Table 1. Status and Control Register Addressing Space

Control Register Name	Address (Hexadecimal)
Bus Timing and Control	Control 02
Bus Timing and Initialization	Control FF
Cache Control[1]	Control 12
Interrupt Status	Control 16
Interrupt/Trap Vector Table	Control 06
I/O Page Register	Control 08
Local Address Register[2]	Control 14
Master Status (MSR)	Control 00
Stack Limit	Control 04
Trap Control	Control 10

NOTES:
1. See section on on-chip memory for register description.
2. See section on multiprocessing mode of operation for register description.

Bus Timing and Control Register. This 8-bit register (Figure 8) governs the timing of transactions to high memory addresses and the daisy-chain timing for interrupt requests, as well as the functionality of requests on the various Z800 MPU interrupt request lines. On reset, this register is cleared to all 0s.

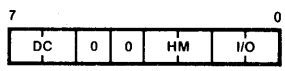


Figure 8. Bus Timing and Control Register

The fields in this register are:

I/O Wait Insertion (I/O). This 2-bit field specifies the number of additional wait states (in addition to the one automatically inserted for I/O) to be inserted by the CPU in both I/O transactions and vector response timing (00 = none, 01 = one, 10 = two, 11 = three).

High Memory Wait Insertion (HM). This 2-bit field specifies the number of automatic wait states (00 = none, 01 = one, 10 = two, 11 = three) for the CPU to insert in memory transactions when the MMU is enabled and there is a 1 in bit 15 of the selected page descriptor register.

Daisy Chain Timing (DC). This 2-bit field determines the number of additional automatic wait states the CPU inserts while the interrupt acknowledge daisy chain is settling (00 = none, 01 = one, 10 = two, 11 = three). A value of 01 in the DC field indicates that one additional cycle will be added to the four cycles that normally elapse between interrupt acknowledge, $\overline{AS}$ and $\overline{DS}$ assertions.

Bus Timing and Initialization Register. This 8-bit register (Figure 9) is used to specify the duration of control signals for the external bus when the MMU is disabled or when the MMU is enabled and there is a 0 in bit 15 of the selected page descriptor register. It also controls the relationship between internal processor clock rates and bus timing. It can be programmed by external hardware upon reset.

During reset this register is initialized to one of two settings, depending on the state of the Wait input line on the rising edge of reset: if the Wait line is not asserted, the register is set to 00_H. If the Wait line is asserted during reset, then this register is set to the contents of the AD lines.

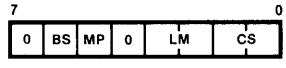


Figure 9. Bus Timing and Initialization Register

The fields in this register are:

Clock Scaling (CS). This 2-bit field specifies the scaling of the CPU clock for all bus transactions (00 = one bus clock cycle is equal to two internal processor clock cycles, 01 = bus clock cycle is equal to the internal processor clock cycle, 10 = one bus clock cycle is equal to four internal processor clock cycles, 11 = reserved). This field cannot be modified by software.

Low Memory Wait Insertion (LM). This 2-bit field specifies the number of automatic wait states (00 = none, 01 = one, 10 = two, 11 = three) for the CPU to insert in memory transactions when the MMU is disabled or when the MMU is enabled and there is a 0 in bit 15 of the selected page descriptor register.

Multiprocessor Configuration Enable (MP). This 1-bit field enables the multiprocessor mode of operation (0 = disabled, 1 = enabled). (See the multiprocessor mode section).

Bootstrap Mode Enable (BS). This 1-bit field enables the bootstrap mode of operation (0 = disabled, 1 = enabled). (See the UART section for details about bootstrap mode.)

Interrupt Status Register. This 16-bit register (Figure 10) indicates which interrupt mode is in effect and which interrupt sources have interrupt requests pending. It also contains the bits that specify whether the interrupt inputs are to be vectored. Only the interrupt vector enable bits are writeable; all other bits are read-only.

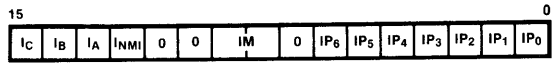


Figure 10. Interrupt Status Register

The fields in this register are:

Interrupt Request Pending (IP). When bit IP_n is set to 1, an interrupt request from sources at level n is pending. (See the Interrupt and Trap Structure section.)

Interrupt Mode (IM). A value of n in this 2-bit field indicates that interrupt mode n is in effect. This field can be changed by executing the IM instruction.

Interrupt Vector Enable (I). These four bits indicate whether each of the four interrupt inputs are to be vectored. When I_n is set to 1, interrupts on the Interrupt n line are vectored when the CPU is in interrupt mode 3; when cleared to 0, all interrupts on this line use the same entry in the Interrupt/Trap Vector Table. These bits are ignored except in interrupt mode 3.

Interrupt/Trap Vector Table Pointer. This 16-bit register (Figure 11) contains the most significant 12 bits of the physical address at the beginning of the Interrupt/Trap Vector Table; the lower 12 bits of the physical address are assumed to be 0. The four least significant bits of this register must be 0.

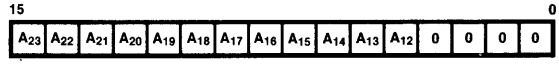


Figure 11. Interrupt/Trap Vector Table Pointer

I/O Page Register. This 8-bit register (Figure 12) indicates the bits to be appended to the 16 bits that are output during I/O transactions during the I/O address phase.

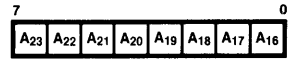


Figure 12. I/O Page Register

Master Status Register. The Master Status register (Figure 13) is a 16-bit register containing status information about the currently-executing program. This register is cleared to 0 during reset.

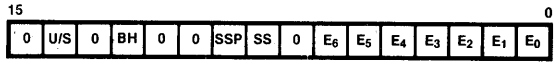


Figure 13. Master Status Register

The fields in this register are:

Interrupt Request Enable (E_n). There are seven Interrupt Enable bits, one for each type of maskable interrupt source (both external and internal). When bit E_n is set to 1, interrupt requests from sources at level n are accepted by the CPU; when this bit is cleared to 0, interrupt requests at level n are not accepted.

Single-Step (SS). While this bit is set to 1, the CPU is in single-stepping mode; while this bit is cleared to 0, automatic single-stepping is disabled. This bit is automatically cleared when a trap or interrupt is taken.

Single-Step Pending (SSP). While this bit is set to 1, the CPU generates a trap prior to executing an instruction. The SS bit is automatically copied into this field at the completion of each instruction. This bit is automatically cleared to 0 when a Single-Step, Page Fault, Privileged Instruction, Break-on-Halt or Division trap is taken so that the SSP bit in the saved Master Status register is cleared to 0.

Breakpoint-on-Halt Enable (BH). While this bit is set to 1, the CPU generates a Breakpoint trap whenever a halt instruction is encountered; while this bit is cleared to 0, the halt instruction is executed normally.

User/System Mode (U/S). While this bit is cleared to 0, the CPU is in the system mode of operation; while it is set to 1 the CPU is in the user mode of operation.

System Stack Limit Register. This 16-bit register (Figure 14) indicates when a System Stack Overflow Warning trap is to be generated. If enabled by setting a control bit in the Trap Status register, pushes onto the system stack cause the 12 most significant bits in this register to be compared to the upper 12 bits of the system Stack Pointer and a trap is generated if they match. The low-order four bits of this register must be 0.

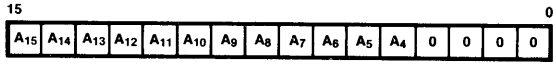


Figure 14. System Stack Limit Register

Trap Control Register. This 8-bit register (Figure 15) enables the maskable traps.

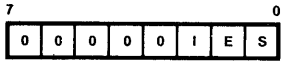


Figure 15. Trap Control Register

The bits in this register are:

System Stack Overflow Warning (S). While this bit is set to 1 the CPU generates a Stack Overflow Warning trap when the system stack enters the specified region of memory. Upon reset this register is initialized to all 0s.

EPU Enable (E). While this bit is cleared to 0, the CPU generates a trap whenever an EPA instruction is encountered.

Inhibit User I/O (I). While this bit is set to 1, the CPU generates a Privileged Instruction trap when an I/O instruction is encountered in user mode.

Cache Control and Local Address Registers. See the on-chip memory section for information about the Cache Control register, and the multiprocessor mode section for information about the Local Address register.

Interrupt and Trap Structure

The Z800 MPU provides a very flexible and powerful interrupt and trap structure. Interrupts are external asynchronous events requiring CPU attention, and are generally triggered by peripherals needing service. Traps are synchronous events resulting from the execution of certain instructions.

Interrupts. Two types of interrupt, nonmaskable and maskable, are supported by the Z800 MPU. The nonmaskable interrupt ($\overline{NMI}$) cannot be disabled (masked) by software and is generally reserved for highest priority external events that require immediate attention. Maskable interrupts, however, can be selectively disabled by software. Both nonmaskable and maskable interrupts can be programmed to be vectored or nonvectored. The CPU accepts interrupts between instructions with the exception of the block move, search, and I/O instructions, which can be safely interrupted after any iteration and restarted after the interrupt is serviced.

Interrupt Sources. The Z800 MPU accepts nonmaskable interrupts on the $\overline{NMI}$ pin only. The Z800 MPU accepts maskable interrupts on the $\overline{INT}$ pins, and from the on-chip counter/timers, DMA channels, and the UART receiver and transmitter. The 40-pin members of the Z800 family accept maskable interrupts on $\overline{INT_A}$ only.

Interrupt Lines A, B, and C can be selectively programmed to support vectored interrupts by setting the appropriate bits in the Interrupt Status register. The external interrupts can be programmed to be vectored or nonvectored in interrupt mode 3.

Interrupt Modes of Operation. The CPU has four modes of interrupt handling. The first three modes extend the Z80 interrupt modes to accommodate additional interrupt input lines in a compatible fashion. The fourth mode provides more flexibility in handling the interrupts. On-chip peripherals use the fourth mode regardless of which mode is selected for externally generated interrupt requests. The interrupt mode is selected by using the privileged instructions IM 0, IM 1, IM 2, or IM 3. On reset, the Z800 MPU is automatically set to interrupt mode 0. The current interrupt mode in effect can be read from the Interrupt Status register.

Mode 0. This mode is identical to the 8080 interrupt response mode. With this mode, the interrupting device on any of the maskable interrupt lines can place a call or restart instruction on the data bus and the CPU will execute it. As a result, the interrupting device, instead of the memory, provides the next instruction to be executed.

Mode 1. When this mode is selected, the CPU responds to a maskable external interrupt by executing a restart to the logical address 0038_H in the system program address space.

Mode 2. This mode is a vectored interrupt response mode. With a single 8-bit byte from the interrupting device, an indirect call can be made to any memory location. With this mode the system maintains a table of 16-bit starting addresses for every interrupt service routine. This table can be located anywhere in the system mode logical data address space on a 256-byte boundary. When an interrupt is accepted, a 16-bit pointer is formed to obtain the desired interrupt service routine starting address from the table. The upper eight bits of this pointer are formed from the contents of the I register. The lower eight bits of the pointer must be supplied by the interrupting device. The 16-bit pointer so formed is treated as a logical address in the system data address space, which can be translated by the MMU to a physical address.

Mode 3. This is the intended mode of operation for systems that take advantage of the enhancements of the Z800 microprocessor family (such as single-step and user/system mode) since the Master Status register is automatically saved and another loaded for the interrupts. Also, vector tables can be used for the external interrupt sources to provide more interrupt vectors for the Z8000™ family, Z80 family, and Z8500 Universal Peripherals.

When an interrupt request (either maskable or nonmaskable) is accepted, the Master Status register, the address of the next instruction to be executed, and a 16-bit "reason code" are pushed onto the system stack. A new Master Status register and Program Counter are then fetched from the Interrupt/Trap Vector Table. The "reason code" for externally generated interrupts is the contents of the bus during the interrupt acknowledge sequence; for 8-bit data buses, the most significant byte of the reason code is zero. For interrupts generated by on-chip peripherals, the reason code identifies which peripheral generated the interrupt and is identical to the vector address in the Interrupt/Trap Vector Table. The Interrupt/Trap Vector Table Pointer is used to reference the table.

Traps. The Z800 CPU supports eight traps that are generated internally. The following traps can be disabled: the EPA trap, which allows software to emulate an EPU; the Stack Warning trap, which is taken at the end of an instruction causing the trap; the Breakpoint-on-Halt trap, which is taken when a halt instruction is encountered; and the Single-Step trap, which is taken for each instruction. In addition, I/O instructions can be specified as privileged instructions. Traps cause the instruction to be terminated without altering CPU registers (except for the system Stack Pointer, which is modified when the program status is pushed onto the system stack).

The saving of the program status on the system stack and the fetching of a new program status from the Interrupt/Trap Vector Table is the same in any interrupt mode of operation.

Traps can only occur if the trap generating features of the Z800 CPU (such as System Stack Overflow warning) have been explicitly enabled. Traps cannot occur on instructions of the Z80 instruction set unless explicitly enabled by the operating system using Z800 CPU extensions.

Extended Instruction. This trap occurs when the CPU encounters an extended instruction while the Extended Processing Architecture (EPA) bit in the Trap Control register is 0. Four trap vectors are used by the EPA trap—one for each type of EPA instruction. This greatly simplifies trap handlers that use I/O instructions to access an EPU or software to emulate an EPU.

Privileged Instruction. This trap occurs whenever an attempt is made to execute a privileged instruction while the CPU is in user mode (User/System Mode control bit in the Master Status register is 1).

System Call. This trap occurs whenever a System Call (SC) instruction is executed.

Access Violation. This trap occurs whenever the MMU's translation mode is enabled and an address to be translated is invalid or (for writes) is write-protected.

System Stack Overflow Warning. This trap occurs only while the Stack Overflow Warning bit in the Trap Control register is set to 1. For each system stack push operation, the most significant bits in the Stack Pointer register are compared with the contents of the Stack Limit register and a trap is signaled if they match. The Stack Overflow Warning bit is then automatically cleared in order to prevent repeated traps.

Division Exception. This trap occurs whenever the divisor is zero (divide-by-zero case) or the true quotient cannot be represented in the destination precision (overflow); the CPU flags are set to distinguish these two cases.

Single-Step. This trap occurs before executing an instruction if the Single-Step Pending control bit in the Master Status register is set to 1. Two control bits in the Master Status register are used for the Single-Step trap. The Single-Step bit (bit 8), on being set when previously clear, causes a trap to occur after the execution of the next instruction. While this bit is set to 1, if an instruction execution causes a trap, the Single-Step trap occurs after the execution of the trap-handling routine. The Single-Step Pending bit (bit 9), is used by the processor to ensure that only one Single-Step trap occurs for each instruction executed while the Single-Step bit is set to 1.

Breakpoint-on-Halt. This trap occurs whenever the Breakpoint-on-Halt control bit in the Master Status register is 1 and a halt instruction is encountered.

Interrupt and Trap Disabling. Maskable interrupts can be enabled or disabled independently via software by setting or clearing the appropriate control bits in the Master Status register.

A 7-bit mask field in the Master Status register indicates which of the requested interrupts will be accepted. Interrupt requests are grouped as follows, with each group controlled by a separate Interrupt Enable control bit. The list is presented in order of decreasing priority, with sources within a group listed in order of descending priority.

- Maskable Interrupt A line (bit 0)
- Counter/Timer 0, DMA0 (bit 1)
- Maskable Interrupt B line (bit 2)
- Counter/Timer 1, UART receiver, DMA1 (bit 3)
- Maskable Interrupt C line (bit 4)
- Counter/Timer 2, UART transmitter, DMA2 (bit 5)
- Counter/Timer 3, DMA3 (bit 6)

When a source of interrupts has been disabled, the CPU ignores any interrupt request from that source.

The System Stack Overflow Warning trap, I/O instructions in user mode trap (Privileged Instruction trap), or Extended Instruction trap can be enabled by setting control bits in the Trap Control register, and the Single-Step and Breakpoint-on-Halt trap can be enabled by setting control bits in the Master Status register; these are the only traps that can be disabled.

Interrupt/Trap Vector Table. The format of the Interrupt/Trap Vector Table consists of pairs of Master Status register and Program Counter words, one pair for each separate on-chip interrupt or trap source. For each external interrupt, there is a separate Master Status register word and Program Counter word (for use if the input is not vectored). If the external interrupt is vectored, a vector table consisting of one Program Counter word for each of the 128 possible vectors that can be returned for each input line is used instead of the dedicated Program Counter word; thus for vectored interrupts, there is only one Master Status register for each interrupt type.

The format of the Interrupt/Trap Vector Table is shown in Table 2.

Table 2. Interrupt/Trap Vector Table

Address (Hexadecimal)	Contents
00	Unused
04	NMI Vector
08	Interrupt Line A Vector (End of Process)
0C	Interrupt Line B Vector
10	Interrupt Line C Vector
14	C-T0
18	C-T1
1C	C-T2
20	C-T3
24	DMA0 Vector
28	DMA1 Vector
2C	DMA2 Vector
30	DMA3 Vector
34	UART Receiver Vector
38	UART Transmitter Vector
3C	Single-Step Trap Vector
40	Breakpoint-on-Halt Trap Vector
44	Division Exception Trap Vector
48	Stack Overflow Warning Trap Vector
4C	Page Fault Trap Vector
50	System Call Trap Vector
54	Privileged Instruction Trap Vector
58	EPU ← Memory Trap Vector
5C	Memory ← EPU Trap Vector
60	A ← EPU Trap Vector
64	EPU Internal Operation Trap Vector
68-6C	Reserved
70-16E	128 Program Counters for NMI and Interrupt line A Vectors (MSR from 04 and 08, respectively)
170-26E	128 Program Counters for Interrupt Line B Vectors (MSR from 0C)
270-36E	128 Program Counters for Interrupt Line C Vectors (MSR from 10)

Addressing Modes

Addressing modes (Figure 16) are used by the CPU to calculate the effective address of an operand needed for execution of an instruction. Nine addressing modes are supported by the Z800 CPU. Of these nine, four are additions to the Z80 addressing modes (Indexed with 16-bit displacement, Stack Pointer Relative, Program Counter Relative, and Base Index) and the remaining five modes are either existing or extensions to the existing Z80 addressing modes.

Register. The operand is one of the 8-bit registers (A, B, C, D, E, H, L, IXH, IHL, IYH or IYL); or one of the 16-bit registers (BC, DE, HL, IX, IY, or SP), or one of the special byte registers (I or R).

Immediate. The operand is in the instruction itself and has no effective address.

Register Indirect. The contents of a register specify the effective address of an operand. The HL register is the register most often used for memory accesses. The C register is used for I/O and control register space accesses.

Direct Address. The effective address of the operand is the location whose address is contained in the instruction. Depending on the instruction, the specified operand is either in I/O or data memory space.

Index. The effective address of the operand is the location specified by adding the 16-bit address contained in the instruction to a twos complement "index" contained in the HL, IX, or IY register.

Short Index. The effective address of the operand is the location computed by adding the 8-bit twos complement signed displacement contained in the instruction to the contents of the IX or IY register. This addressing mode is equivalent to the Z80 CPU indexed mode.

Relative. An 8- or 16-bit displacement contained in the instruction is added to the Program Counter to generate the effective address of the operand.

Stack Pointer Relative. The effective address of the operand is the location computed by adding a 16-bit twos complement displacement contained in the instruction to the contents of the Stack Pointer.

Base Index. The effective address of the operand is the location whose address is computed by adding the contents of HL, IX, or IY to the contents of another of these three registers.

Instruction Set

Notation

Addressing Modes. The following notation is used to describe the addressing modes and instruction operations as shown in the instruction set.

BX	Base Index
DA	Direct Address
IM	Immediate constant
IR	Indirect Register
X	Index
R	Single register of the set (A, B, C, D, E, H, L)
RA	Relative address
RX	A byte in the IX or IY register
SP	Current Stack Pointer
SR	Stack Relative
SX	Short Index
n	8-bit constant
nn	16-bit constant

Symbols. The following symbols are used to describe the instruction set.

dst	(Destination location or contents)
src	(Source location or contents)
n	(An 8-bit constant)
nn	(A 16-bit constant)
SP	(Current Stack Pointer)
p	(Interrupt mode)
(C)	(I/O port pointed to by C register)
SSP	(System Stack Pointer)
USP	(User Stack Pointer)

Assignment of a value is indicated by the symbol "←". For example,

$$dst \leftarrow dst + src$$

indicates that the source data is added to the destination data and the result is stored in the destination location. The notation "addr(n)" is used to refer to bit "n" of a given location. For example,

$$dst(7)$$

specifies bit 7 of the destination.

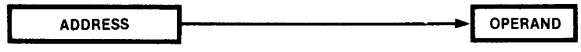
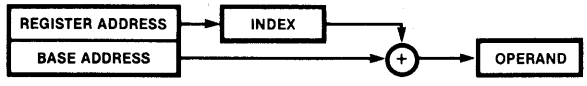
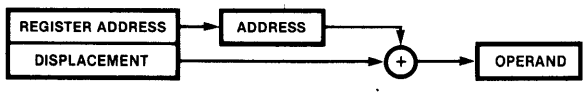
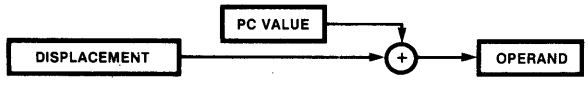
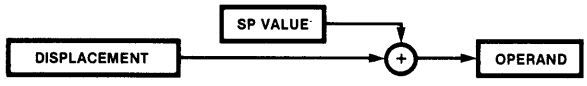
Flags. The F register contains the following six flags.

C	Carry flag
H	Half carry flag
N	Add/Subtract flag
P/V	Parity/Overflow flag
S	Sign flag
Z	Zero flag

Condition Codes. The following symbols describe the condition codes.

Z*	Zero
NZ*	Not zero
C*	Carry
NC*	No carry
S	Sign
NS	No sign
V	Overflow
PE	Parity even
PO	Parity odd
P	Positive
M	Minus

*Abbreviated set

Mode	Operand Addressing			Operand Value
	In the Instruction	In a Register	In Memory or I/O	
Register				The content of the register
Immediate				In the instruction
Register Indirect				The content of the location whose address is in the register
Direct Address				The content of the location whose address is in the instruction
***Index**				The content of the location whose address is the 16-bit address in the instruction, offset by the content of the 16-bit register
Short Index				The content of the location whose address is in the 16-bit register, offset by the 8-bit displacement in the instruction
***Relative**				The content of the location whose address is the content of the Program Counter, offset by the displacement in the instruction
***Stack Pointer Relative**				The content of the location whose address is the content of the Stack Pointer, offset by the displacement in the instruction
***Base Index**				The content of the location whose address is the content of a register, offset by the displacement in a register

*New Z800 Family addressing modes

Figure 16. Addressing Modes

8-Bit Load Group

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
EX A,src	src = R,RX,IR,DA,X,SX, RA,SR,BX	•	•	•	•	•	•	**Exchange Accumulator** A ↔ src
EX H,L		•	•	•	•	•	•	**Exchange H,L** H ↔ L
LD dst,src	src = A dst = R,RX,IR,DA,X, SX,RA,SR,BX, (BC),(DE) or src = R,RX,IM,IR,DA, X,SX,RA,SR,BX, (BC),(DE) dst = A	•	•	•	•	•	•	**Load Accumulator** dst ← src
LD dst,src	dst = R src = R,RX†,IM,IR,SX or dst = R,RX†,IR,SX src = R	•	•	•	•	•	•	**Load Register (Byte)** dst ← src
LD dst,n	dst = R,RX,IR,DA,X, SX,RA,SR,BX	•	•	•	•	•	•	**Load Immediate (Byte)** dst ← nn
* **LDUD** dst,src	dst = A src = IR or SX in user space or dst = IR or SX in user space src = A	•	‡	•	‡	•	‡	**Load in User Data Space (Byte)** dst ← src
* **LDUP** dst,src	dst = A src = IR or SX in user space or dst = IR or SX in user space src = A	•	‡	•	‡	•	‡	**Load in User Program Space (Byte)** dst ← src

16-Bit Load Group

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
EX src,HL	src = DE,IX,IY	•	•	•	•	•	•	**Exchange HL with Addressing Register** src ↔ HL

* Privileged instruction.
† Accessing bytes of IX or IY precludes use of H or L.

16-Bit Load Group (Continued)

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
EX (SP),dst	dst = HL,IX,IY	•	•	•	•	•	•	**Exchange Addressing Register with Top of Stack** *(SP)* ↔ *dst*
EX AF,AF′		↕	↕	↕	↕	↕	↕	**Exchange Accumulator/ Flag with Alternate Bank** AF ↔ AF′
EXX		•	•	•	•	•	•	**Exchange Byte/Word Registers with Alternate Bank** BC ↔ BC′ DE ↔ DE′ HL ↔ HL′
LD[W] dst,src	dst = HL,IX,IY src = IM,DA,X,RA,SR,BX or dst = DA,X,RA,SR,BX src = HL,IX,IY	•	•	•	•	•	•	**Load Addressing Register** dst ← src
LD[W] dst,src	dst = BC,DE,HL,SP src = IM,IR,DA,SX or dst = IR,DA,SX src = BC,DE,HL,SP	•	•	•	•	•	•	**Load Register Word** dst ← src
LDW dst,nn	dst = RR,IR,DA,RA	•	•	•	•	•	•	**Load Immediate Word** dst ← nn
LD[W] dst,nn	dst = RR	•	•	•	•	•	•	**Load Immediate Word** dst ← nn
LD[W] dst,src	dst = SP src = HL,IX,IY,IM,IR, DA,SX or dst = IR,DA,SX src = SP	•	•	•	•	•	•	**Load Stack Pointer** dst ← src
LDA dst,src	dst = HL,IX,IY src = X,RA,SR,BX	•	•	•	•	•	•	**Load Address** dst ← address (src)
POP dst	dst = RR*,IR,DA,RA	•	•	•	•	•	•	**POP** dst ← (SP) SP ← SP + 2
PUSH src	src = RR*,IM,IR,DA, RA	•	•	•	•	•	•	**PUSH** SP ← SP − 2 (SP) ← src

*AF instead of SP.

Block Transfer and Search Group

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
CPD		↕	↕	↕	↕	1	•	**Compare and Decrement** A − (HL) HL ← HL − 1 BC ← BC − 1
CPDR		↕	↕	↕	↕	1	•	**Compare, Decrement and Repeat** Repeat until BC = 0 or match: A − (HL) HL ← HL − 1 BC ← BC − 1
CPI		↕	↕	↕	↕	1	•	**Compare and Increment** A − (HL) HL ← HL + 1 BC ← BC − 1
CPIR		↕	↕	↕	↕	1	•	**Compare, Increment and Repeat** Repeat until BC = 0 or match: A − (HL) HL ← HL + 1 BC ← BC − 1
LDD		•	•	0	↕	0	•	**Load and Decrement** (DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1
LDDR		•	•	0	0	0	•	**Load, Decrement and Repeat** Repeat until BC = 0: (DE) ← (HL) DE ← DE − 1 HL ← HL − 1 BC ← BC − 1
LDI		•	•	0	↕	0	•	**Load and Increment** (DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1
LDIR		•	•	0	0	0	•	**Load, Increment and Repeat** Repeat until BC = 0: (DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC − 1

8-Bit Arithmetic and Logic Group

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
ADC [A,]src	src = R,RX,IM,IR, DA,X,SX,RA, SR,BX	↕	↕	↕	V	0	↕	**Add With Carry (Byte)** A ← A + src + C
ADD [A,]src	src = R,RX,IM,IR, DA,X,SX,RA, SR,BX	↕	↕	↕	V	0	↕	**Add (Byte)** A ← A + src
AND [A,]src	src = R,RX,IM,IR, DA,X,SX,RA, SR,BX	↕	↕	1	P	0	0	**And** A ← A AND src
CP [A,]src	src = R,RX,IM,IR, DA,X,SX,RA, SR,BX	↕	↕	↕	V	1	↕	**Compare (Byte)** A − src
CPL [A]		•	•	1	•	1	•	**Complement Accumulator** A ← NOT A
DAA [A]		↕	↕	↕	P	•	↕	**Decimal Adjust Accumulator** A ← Decimal Adjust A
DEC dst	dst = R,RX,IR,DA,X, SX,RA,SR,BX	↕	↕	↕	V	1	•	**Decrement (Byte)** dst ← dst − 1
DIV [HL,]src	src = R,RX,IM,DA,X, SX,RA,SR,BX	↕	↕	•	↕	•	•	**Divide (Byte)** A ← HL ÷ src L ← remainder
DIVU [HL,]src	src = R,RX,IM,DA,X, SX,RA,SR,BX	0	↕	•	↕	•	•	**Divide Unsigned (Byte)** A ← HL ÷ src L ← remainder
EXTS [A]		•	•	•	•	•	•	**Extend Sign (Byte)** L ← A If A(7) = 0, then H ← 00 else H ← FF
INC dst	dst = R,RX,IR,DA,X, SX,RA,SR,BX	↕	↕	↕	V	0	•	**Increment (Byte)** dst ← dst + 1
MULT [A,]src	src = R,RX,IM,IR,DA, X,SX,RA,SR,BX	↕	↕	•	0	•	↕	**Multiply (Byte)** HL ← A × src
MULTU [A,]src	src = R,RX,IM,IR,DA, X,SX,RA,SR,BX	0	↕	•	0	•	↕	**Multiply Unsigned (Byte)** HL ← A × src
NEG [A]		↕	↕	↕	V	1	↕	**Negate Accumulator** A ← −A

8-Bit Arithmetic and Logic Group (Continued)

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
OR [A,]src	src = R,RX,IM,IR,DA, X,SX,RA,SR,BX	↕	↕	0	P	0	0	**OR** A ← A OR src
SBC [A,]src	src = R,RX,IM,IR,DA, X,SX,RA,SR,BX	↕	↕	↕	V	1	↕	**Subtract With Carry (Byte)** A ← A − src − c
SUB [A,]src	src = R,RX,IM,IR,DA, X,SX,RA,SR,BX	↕	↕	↕	V	1	↕	**Subtract** A ← A − src
XOR [A,]src	src = R,RX,IM,IR,DA, X,SX,RA,SR,BX	↕	↕	0	P	0	0	**Exclusive OR** A ← A XOR src

16-Bit Arithmetic Operations

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
ADC dst,src	dst = HL src = BC,DE,HL,SP or dst = IX src = BC,DE,IX,SP or dst = IY src = BC,DE,IY,SP	↕	↕	↕	V	0	↕	**Add With Carry (Word)** dst ← dst + src + c
ADD dst,src	dst = HL src = BC,DE,HL,SP or dst = IX src = BC,DE,IX,SP or dst = IY src = BC,DE,IY,SP	•	•	↕	•	0	↕	**Add (Word)** dst ← dst + src
ADD dst,A	dst = HL,IX,IY	↕	↕	↕	V	0	↕	**Add Accumulator to Addressing Register** dst ← dst + A
ADDW [HL,]src	src = RR*,IM,DA,X,RA	↕	↕	↕	V	0	↕	**Add Word** HL ← HL + src
CPW [HL,]src	src = RR*,IM,DA,X,RA	↕	↕	↕	V	1	↕	**Compare (Word)** HL − src
DECW dst	dst = RR*,IR,DA,X,RA	•	•	•	•	•	•	**Decrement (Word)** dst ← dst − 1

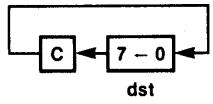
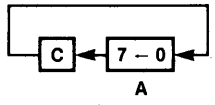
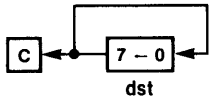
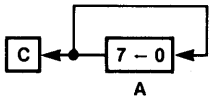
*In X addressing mode, (HL + nn) is precluded.

16-Bit Arithmetic Operations (Continued)

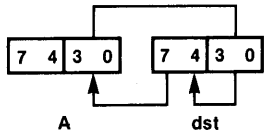
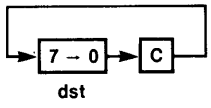
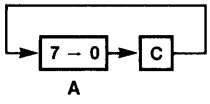
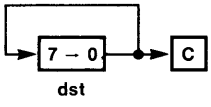
Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
DEC[W] dst	dst = RR	•	•	•	•	•	•	**Decrement (Word)** dst ← dst − 1
DIVUW [DEHL,]src	src = RR,IM,DA,X,RA	0	↕	•	↕	•	•	**Divide Unsigned (Word)** HL ← DEHL ÷ src DE ← remainder
DIVW [DEHL,]src	src = RR,IM,DA,X,RA	↕	↕	•	↕	•	•	**Divide (Word)** HL ← DEHL ÷ src DE ← remainder
EXTS HL		•	•	•	•	•	•	**Extend Sign (Word)** If H(7) = 0, then DE ← 0000 else DE ← FFFF
INCW dst	dst = RR,IR,DA,X*,RA	•	•	•	•	•	•	**Increment (Word)** dst ← dst + 1
INC[W] dst	dst = RR	•	•	•	•	•	•	**Increment (Word)** dst ← dst + 1
MULTUW [HL,]src	src = RR,IM,DA,X,RA	0	↕	•	0	•	↕	**Multiply Unsigned (Word)** DEHL ← HL × src
MULTW [HL,]src	src = RR,IM,DA,X,RA	↕	↕	•	0	•	↕	**Multiply (Word)** DEHL ← HL × src
NEG HL		↕	↕	↕	V	1	↕	**Negate HL** HL ← −HL
SBC dst,src	dst = HL src = BC,DE,HL,SP or dst = IX src = BC,DE,IX,SP or dst = IY src = BC,DE,IY,SP	↕	↕	↕	V	1	↕	**Subtract With Carry (Word)** dst ← dst − src − C
SUBW [HL,]src	src = RR,IM,DA,X*,RA	↕	↕	↕	V	1	↕	**Subtract (Word)** HL ← HL − src

*In X addressing mode, (HL + nn) is precluded.

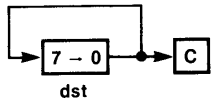
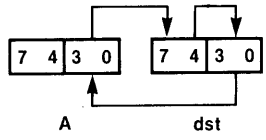
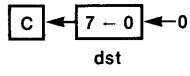
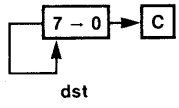
Bit Manipulation, Rotate and Shift Group

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
BIT b,dst	dst = R,IR,SX	•	‡	1	•	0	•	**Bit Test** Z ← NOT dst(b)
RES b,dst	dst = R,IR,SX	•	•	•	•	•	•	**Reset bit** dst(b) ← 0
RL dst	dst = R,IR,SX	‡	‡	0	P	0	‡	**Rotate Left** tmp ← dst dst(0) ← C C ← dst(7) dst(n + 1) ← tmp(n) for n = 0 to 6 
RLA		•	•	0	•	0	‡	**Rotate Left Accumulator** tmp ← A A(0) ← C C ← A(7) A(n + 1) ← tmp(n) for n = 0 to 6 
RLC dst	dst = R,IR,SX	‡	‡	0	P	0	‡	**Rotate Left Circular** tmp ← dst C ← dst(7) dst(0) ← tmp(7) dst(n + 1) ← tmp(n) for n = 0 to 6 
RLCA		•	•	0	•	0	‡	**Rotate Left Circular (Accumulator)** tmp ← A C ← A(7) A(0) ← tmp(7) A(n + 1) ← tmp(n) for n = 0 to 6 

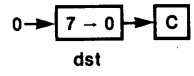
Bit Manipulation, Rotate and Shift Group (Continued)

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
RLD		↕	↕	0	P	0	•	**Rotate Left Digit** tmp(0:3) ← A(0:3) A(0:3) ← src(4:7) src(4:7) ← src(0:3) src(0:3) ← tmp(0:3) 
RR dst	dst = R,IR,SX	↕	↕	0	P	0	↕	**Rotate Right** tmp ← dst dst(7) ← C C ← dst(0) dst(n) ← tmp(n + 1) for n = 0 to 6 
RRA		•	•	0	•	0	↕	**Rotate Right (Accumulator)** tmp ← dst A(7) ← C C ← A(0) A(n) ← tmp(n + 1) for n = 0 to 6 
RRC dst	dst = R,IR,SX	↕	↕	0	P	0	↕	**Rotate Right Circular** tmp ← dst C ← dst(0) dst(7) ← tmp(0) dst(n) ← tmp(n + 1) for n = 0 to 6 

Bit Manipulation, Rotate and Shift Group (Continued)

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
RRCA		•	•	0	•	0	↕	**Rotate Right Circular (Accumulator)** tmp ← A C ← A(0) A(7) ← tmp(0) A(n) ← tmp(n + 1) for n = 0 to 6 
RRD dst	dst = IR	↕	↕	0	P	0	•	**Rotate Right Digit** tmp(0:3) ← A(0:3) A(0:3) ← src(0:3) src(0:3) ← src(4:7) src(4:7) ← tmp(0:3) 
SET b,dst	dst = R,IR,SX	•	•	•	•	•	•	**Set Bit** dst(b) ← 1
SLA dst	dst = R,IR,SX	↕	↕	0	P	0	↕	**Shift Left Arithmetic** tmp ← dst C ← dst(7) dst(0) ← 0 dst(n + 1) ← tmp(n) for n = 0 to 6 
SRA dst	dst = R,IR,SX	↕	↕	0	P	0	↕	**Shift Right Arithmetic** tmp ← dst C ← dst(0) dst(7) ← tmp(7) dst(n) ← tmp(n + 1) for n = 0 to 6 

Bit Manipulation, Rotate and Shift Group (Continued)

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
SRL dst	dst = R,IR,SX	0	↕	0	P	0	↕	**Shift Right Logical** tmp ← dst C ← dst(0) dst(7) ← 0 dst(n) ← tmp(n + 1) for n = 0 to 6  dst
TSET dst	dst = R,IR,SX	↕	•	•	•	•	•	**Test and Set** s ← dst(7) dst ← FF

Program Control Group

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
CALL cc,dst	dst = IR,DA,RA	•	•	•	•	•	•	**CALL** If cc is satisfied then: SP ← SP − 2 (SP) ← PC PC ← dst
CALL dst	dst = IR,DA,RA	•	•	•	•	•	•	**CALL** SP ← SP − 2 (SP) ← PC PC ← dst
CCF		•	•	↕	•	0	↕	**Complement Carry Flag** C ← NOT C
DJNZ dst	dst = RA	•	•	•	•	•	•	**Decrement and Jump if Non-Zero** B ← B − 1 If B≠0 then PC ← dst
JAF dst	dst = RA	•	•	•	•	•	•	**Jump on Auxiliary Accumulator/Flag** If Auxiliary AF then: PC ← dst
JAR dst	dst = RA	•	•	•	•	•	•	**Jump on Auxiliary Register File in Use** If Auxiliary File then: PC ← dst
JP cc,dst	dst = IR,DA,RA	•	•	•	•	•	•	**Jump** If cc is satisfied then: PC ← dst

Program Control Group (Continued)

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
	JP dst	dst = IR,DA,RA	•	•	•	•	•	•	**Jump** PC ← dst
	JR cc,dst	dst = RA	•	•	•	•	•	•	**Jump Relative** If cc* is satisfied then: PC ← PC + dst
	JR dst	dst = RA	•	•	•	•	•	•	**Jump Relative** PC ← PC + dst
	RET		•	•	•	•	•	•	**Return** PC ← (SP) SP ← SP − 2
	RET cc		•	•	•	•	•	•	**Return** If cc is satisfied then: PC ← (SP) SP ← SP + 2
	RST dst†	dst = DA	•	•	•	•	•	•	**Restart** SP ← SP − 2 (SP) ← PC PC ← dst
	SC nn		•	•	•	•	•	•	**System Call** SP ← SP − 4 (SP) ← PS SP ← SP − 2 (SP) ← nn PS ← System Call Program Status
	SCF		•	•	0	•	0	1	**Set Carry Flag** C ← 1

Input/Output Instruction Group

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
‡	**IN** dst,(C)	dst = R,RX,DA,X,RA, SR,BX	↕	↕	0	P	0	•	**Input** dst ← (C)
‡	**IN** A,(n)		•	•	•	•	•	•	**Input Accumulator** A ← (n)

* Uses abbreviated set of condition codes.
† dst must be 0, 8, 16, 24, 32, 40, or 56.
‡ Programmable as privileged.

Input/Output Instruction Group (Continued)

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
†	**IN[W]** HL,(C)		•	•	•	•	•	•	**Input HL** HL ← (C)
†	**IND**		•	↕	•	•	1	•	***Input and Decrement*** **(Byte)** (HL) ← (C) B ← B − 1 HL ← HL − 1
†	**INDW**		•	↕	•	•	1	•	**Input and Decrement (Word)** (HL) ← (C) B ← B − 1 HL ← HL − 2
†	**INDR**		•	1	•	•	1	•	**Input, Decrement and Repeat (Byte)** Repeat until B = 0: HL ← (C) B ← B − 1 HL ← HL − 1
†	**INDRW**		•	1	•	•	1	•	**Input, Decrement and Repeat (Word)** Repeat until B = 0: HL ← (C) B ← B − 1 HL ← HL − 2
†	**INI**		•	↕	•	•	1	•	**Input and Increment (Byte)** (HL) ← (C) B ← B − 1 HL ← HL + 1
†	**INIW**		•	↕	•	•	1	•	**Input and Increment (Word)** (HL) ← (C) B ← B − 1 HL ← HL + 2
†	**INIR**		•	1	•	•	1	•	**Input, Increment and Repeat (Byte)** Repeat until B = 0: (HL) ← (C) HL ← HL + 1 B ← B − 1

† Programmable as privileged.

Input/Output Instruction Group (Continued)

Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
† **INIRW**		•	1	•	•	1	•	**Input, Increment and Repeat (Word)** Repeat until B = 0: (HL) ← (C) HL ← HL − 2 B ← B − 1
† **OUT** (C),src	src = R,RX,DA,X,RA, SR,BX	•	•	•	•	•	•	**Output** (C) ← src
† **OUT** (n),A		•	•	•	•	•	•	**Output Accumulator** (n) ← A
† **OUT[W]** (C),HL		•	•	•	•	•	•	**Output HL** (C) ← HL
† **OUTD**		•	↕	•	•	1	•	**Output and Decrement (Byte)** B ← B − 1 (C) ← (HL) HL ← HL − 1
† **OUTDW**		•	↕	•	•	1	•	**Output and Decrement (Word)** B ← B − 1 (C) ← (HL) HL ← HL − 2
† **OUTDR**		•	1	•	•	1	•	**Output, Decrement and Repeat (Byte)** Repeat until B = 0: B ← B − 1 (C) ← (HL) HL ← HL − 1
† **OUTDRW**		•	1	•	•	1	•	**Output, Decrement and Repeat (Word)** Repeat until B = 0: B ← B − 1 (C) ← (HL) HL ← HL − 2
† **OUTI**		•	↕	•	•	1	•	**Output and Increment (Byte)** B ← B − 1 (C) ← (HL) HL ← HL + 1

† Programmable as privileged.

Input/Output Instruction Group (Continued)

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
†	**OUTIW**		•	↕	•	•	1	•	**Output and Increment (Word)** B ← B − 1 *(C) ← (HL)* HL ← HL + 2
†	**OUTIR**		•	1	•	•	1	•	**Output, Increment and Repeat (Byte)** Repeat until B = 0: B ← B − 1 (C) ← (HL) HL ← HL + 1
†	**OUTIRW**		•	1	•	•	1	•	**Output, Increment and Repeat (Word)** Repeat until B = 0: B ← B − 1 (C) ← (HL) HL ← HL + 2
†	**TSTI** (C)		↕	↕	0	P	0	•	**Test Input** F ← test (C)

CPU Control Group

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
*	**DI** Int	Int = E6,E5,E4,E3,E2, E1,E0	•	•	•	•	•	•	**Disable Interrupt** If Ei then: MSR(i) ← 0; Otherwise $MSR_{0-6} \leftarrow 0$
*	**EI** Int	Int = E6,E5,E4,E3,E2, E1,E0	•	•	•	•	•	•	**Enable Interrupt** If Ei then: MSR(i) ← 1; Otherwise $MSR_{0-6} \leftarrow 1$
*	**HALT**		•	•	•	•	•	•	**Halt** CPU Halts
*	**IM** p	p = 0,1,2,3	•	•	•	•	•	•	**Interrupt Mode Select** Interrupt Mode ← p
*	**LD** dst,src	dst = A src = I,R	↕	↕	0	↕	0	•	**Load Accumulator from I or R Register** A ← src

† Programmable as privileged.
* Privileged instruction.

CPU Control Group (Continued)

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
*	**LD** dst,src	dst = I,R src = A	•	•	•	•	•	•	**Load I or R Register from Accumulator** dst ← A
*	**LDCTL** dst,src	dst = (C),USP src = HL,IX,IY or dst = HL,IX,IY src = (C),USP	•	•	•	•	•	•	**Load Control** dst ← src
	NOP		•	•	•	•	•	•	**No Operation**
	PCACHE		•	•	•	•	•	•	**Purge Cache** All cache entries invalidated
*	**RETI**		•	•	•	•	•	•	**Return from Interrupt** PC ← (SP) SP ← SP + 2
*	**RETIL**		•	•	•	•	•	•	**Return from Interrupt Long** PS ← (SP) SP ← SP + 4
*	**RETN**		•	•	•	•	•	•	**Return from Nonmaskable Interrupt** PC ← (SP) SP ← SP + 2 MSR(0-7) ← IFF(0-7)

Extended Instruction Group[1]

	Instruction	Addressing Modes	Flags S	Z	H	P/V	N	C	Operation
	EPUM src	src = IR,DA,X,RA,SR,BX	•	•	•	•	•	•	**Load EPU from Memory** EPU ← template EPU ← src
	MEPU dst	dst = IR,DA,X,RA,SR,BX	•	•	•	•	•	•	**Load Memory from EPU** EPU ← template dst ← EPU
	EPUF		↕	↕	0	P	0	•	**Load Accumulator from EPU** EPU ← template A ← EPU
	EPUI		•	•	•	•	•	•	**EPU Internal Operation** EPU ← template

[1]Refer to the Z8070 Z8000™ Floating-Point Product Specification (document number 00-2235-01) for the floating-point extended instructions.

* Privileged Instruction.

EXTENDED PROCESSING ARCHITECTURE

Features

The Zilog Extended Processing Architecture (EPA) provides an extremely flexible and modular approach to expanding both the hardware and software capabilities of the Z800 CPU. Features of the EPA include:

- Allows Z800 CPU instruction set to be extended by external devices.
- Increases throughput of the system by using up to four specialized external processors in parallel with the CPU.
- Used by Z8070 floating-point EPU.
- Permits modular design of Z800 CPU-based systems.
- Provides easy management of multiple microprocessor configurations via "single instruction stream" communication.
- Simple interconnection between EPUs and Z800 MPU requires no additional external supporting logic.
- Supports debugging of suspect hardware against proven software.
- EPUs can be added as the system grows and as EPUs with specialized functions are developed.

General Description

The processing power of the Zilog Z-BUS Z800 microprocessor can be boosted beyond its intrinsic capability by the Extended Processing Architecture (EPA). The EPA allows the Z800 CPU to accommodate up to four Extended Processing Units (EPUs), which perform specialized functions in parallel with the CPU's main instruction execution stream.

The EPUs connect directly to the Z-BUS and continuously monitor the CPU instruction stream for an instruction intended for the EPU (template). When a template is detected, the appropriate EPU responds, obtaining or placing data or status information on the Z-BUS by using the Z800 CPU-generated control signals and performing its function as directed.

The CPU is responsible for instructing the EPU and delivering operands and data to it. The EPU recognizes templates intended for it and executes them, using data supplied with the template and/or data within its internal registers. There are three classes of EPU instructions:

- Data transfers between main memory and EPU registers
- Data transfers between CPU registers and EPU status registers
- EPU internal operations

Six addressing modes can be utilized with transfers between EPU registers and the CPU and main memory:

- Direct Address
- Indirect Register
- Indexed
- Stack Pointer Relative
- Relative
- Base Index

In addition to the hardware-implemented capabilities of the EPA, there is an extended instruction trap mechanism to permit software simulation of EPU functions. An EPU present bit in the Z800 MPU Trap Control register indicates whether actual EPUs are present or not. If not, when the CPU traps when an extended instruction is detected, a software "trap handler" can emulate the desired EPU function. Thus, the EPA software trap routine supports systems not containing an EPU.

EPA and CPU instruction execution are shown in Figure 17. The CPU begins operation by fetching an instruction and determining whether or not it is an EPU instruction. If the instruction is an EPU instruction, the state of the EPU Enable bit in the Trap Control register is examined. If the EPU Enable bit is reset (E = 0), the CPU generates a trap and the EPU instruction can be simulated by an EPU instruction trap software routine. However, if the EPU Enable bit is set (E = 1), indicating that an EPU is present in the system, then the 4-byte EPU template is fetched from memory. The fetching of the EPU template is indicated by the status lines ST_0–ST_3. The EPU meanwhile continuously monitors the Z-BUS and the status lines for its own templates. After fetching the EPU template, the CPU, if necessary, transfers appropriate data between the CPU and memory or between the CPU and the EPU. These transactions are indicated by unique encodings of the status lines. If the EPU is free when the template and the data appear, the EPU template is executed. If the EPU is still processing a previous instruction, it activates the $\overline{PAUSE}$ line (Z8216 only) to halt further execution of CPU instructions until execution is complete. After the execution of the template is complete, the EPU deactivates the $\overline{PAUSE}$ line and CPU instruction execution continues.

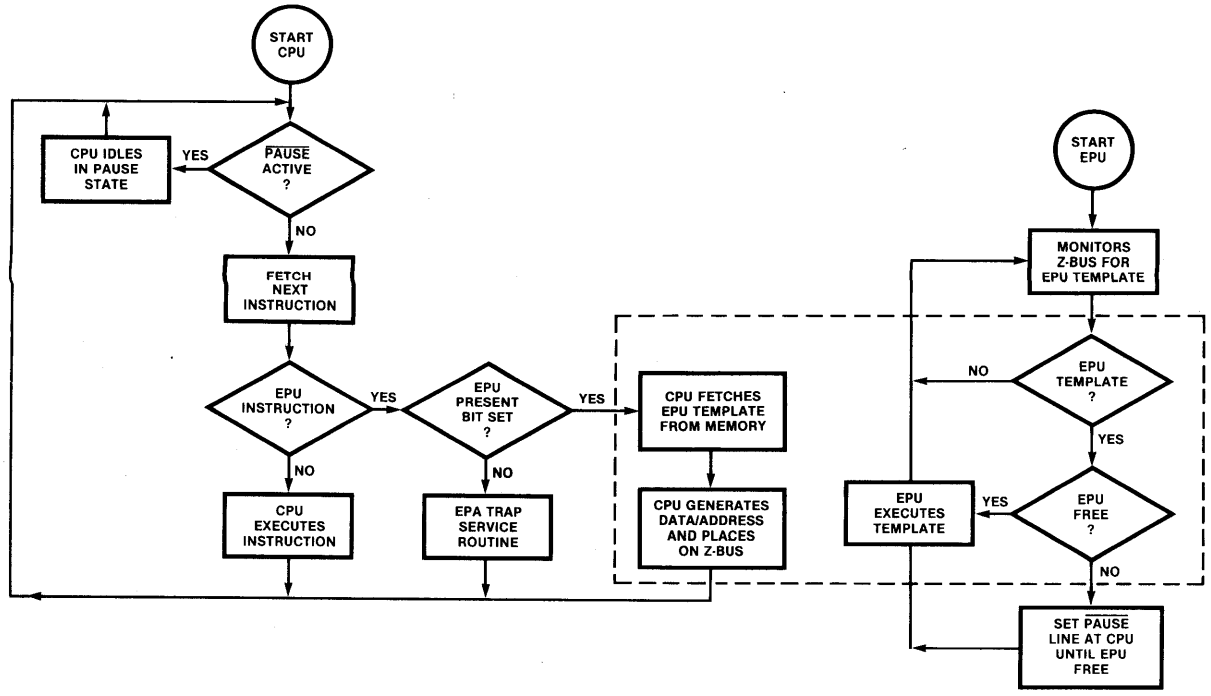


Figure 17. EPA and Z8216 CPU Instruction Execution

MEMORY MANAGEMENT

Features

- On-chip dynamic address translation
- Permits addressing of large physical memory
  - 512K bytes—40-pin devices
  - 16M bytes—64-pin devices
- Separate translation facilities for user and system modes
- Permits instructions and data to reside in separate memory areas.
- Write protection for individual pages of memory
- Aborts CPU on access violation to support virtual memory

General Description

The Z800 microprocessor contains an on-chip Memory Management Unit (MMU), which translates logical addresses into physical addresses. This allows access to more than 64K bytes of physical memory and provides memory protection features typical of those found on large systems. With the MMU, the CPU can access up to 16M bytes of physical memory, depending on package size (the 40-pin package devices output only 19 address bits). The MMU features a sophisticated trapping mechanism that generates page faults on error conditions. Instructions that are aborted by a page fault can be restarted in a manner compatible with virtual memory system requirements. On reset, the MMU features are not enabled, thus permitting logical addresses to pass to the physical memory untranslated.

The physical address space is expanded by dividing the 64K byte logical address space (the space manipulated by the program) into pages. The pages are then mapped (translated) into the larger physical address space of the Z800 microprocessor. The mapping process makes the user software addresses independent of the physical memory, so the user is freed from specifying where information is actually stored in physical memory. The actual size of the page depends on whether the program/data separation mode is enabled—if it is enabled, each page is 8K bytes in length, and if it is not enabled, the page length is 4K bytes. With the page mapping technique, 16-bit logical addresses can be translated into 24-bit physical addresses (only the lower 19 bits are externally available on 40-pin devices). Address translation can occur both in system and in user mode, with separate translation facilities available to each mode. The MMU further allows instruction references to be separated from data references, which enables programs of up to 64K bytes in length to manipulate up to 64K bytes of data without operating system intervention.

MMU Architecture

The Z800 MMU consists of two sets of sixteen page descriptor registers (Figure 18) that are used to translate addresses, a 16-bit control register that governs the translation facilities, a Page Descriptor Register Pointer, an I/O write-only port that can be used to invalidate sets of page descriptors, and two I/O ports for accesses to the page descriptor registers. One set of page descriptor registers is dedicated to the system mode of operation and the other set is dedicated to the user mode of operation.

While an address is being translated, attributes associated with the logical page containing that location are checked. The correct logical page is determined by the CPU mode (user or system), address space (program/data), and the four most significant bits of the logical address. Pages can be write-protected to prevent them from being modified by the executing task and can also be marked as non-cacheable to prevent information from being copied into the cache for later reference. The latter capability is useful in multiprocessor systems, to ensure that the processor always accesses the most current version of information being shared among multiple devices. The MMU also maintains a bit for each page that indicates if the page has been modified.

Each page descriptor register contains a Valid bit, which indicates that the descriptor contains valid information. Any attempt by the MMU to translate an address using an invalid descriptor generates a page fault. Valid bits for groups of page descriptor registers can be reset by writing to an MMU control port.

Figure 18. Page Descriptor Register

For each mode of CPU operation, the MMU can be configured to separate instruction fetches from data fetches, and thus separate the program address space from the data address space. When the program/data separation mode is in effect, the sixteen page descriptor registers for the current CPU mode of operation (user or system) are partitioned into two sets, one for instruction fetches and one for data fetches. A instruction fetch or data access using the Program Counter Relative addressing mode is translated by the MMU registers associated with the program address space; data accesses using other addressing modes and accesses to the Interrupt Vector Table in interrupt mode 2 use the MMU registers associated with the data address space. In this mode of MMU operation, the page size is 8192 bytes. There are two control bits in the MMU Master Control register that independently specify whether the user and system modes of CPU operation have separate program and data address spaces.

Each 16-bit page descriptor register consists of a 4-bit attribute field and a 12-bit page frame address field. The attribute field consists of the least significant bits of the descriptor and contains four control and status bits, listed below.

Modified (M). This bit is automatically set whenever a write is successfully performed to a logical address in this page; it can be cleared to 0 only by a software routine that loads the descriptor register. If the Valid bit is 0, the contents of this bit are undefined.

Cacheable (C). While this bit is set to 1, information fetched from this page can be placed in the cache. While this bit is cleared to 0, the cache control mechanism is inhibited from retaining a copy of the information.

Write-Protect (WP). While this bit is set to 1, CPU writes to logical addresses in this page cause a page fault to be generated and prevent a write operation from occurring. While this bit is cleared to 0, all valid accesses are permitted.

Valid (V). While this bit is set to 1, the descriptor contains valid information. While this bit is cleared to 0, all CPU accesses to logical addresses in this page cause a page fault to be generated.

MMU Control Registers and I/O Ports

MMU operation is controlled by one control register and four dedicated I/O ports. The MMU Master Control register (Figure 19) determines the program/data address space separation in effect in both user and system modes and whether logical addresses generated in user and system mode will be translated by the MMU. Page descriptor registers are accessed indirectly through the register address contained in the Page Descriptor Register Pointer. The descriptor select port is used to access the page descriptor register that is pointed to by the Page Descriptor Register Pointer. After this access the Page Descriptor Register Pointer is left unchanged. The block move I/O port is used to move blocks of words between the page descriptor registers and memory; reads or writes to this I/O port access data pointed to by the Page Descriptor Register Pointer, then increment the pointer by one. The Invalidation I/O Port is used to invalidate blocks of page descriptor registers; writes to this port cause the Valid bits in selected blocks of page descriptor registers to be cleared to 0, which indicates that the descriptors no longer contain valid information.

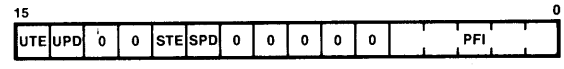


Figure 19. MMU Master Control Register

MMU Master Control Register. The MMU Master Control register controls the operation of the MMU. This register contains four control bits; all other bits in this

register must be cleared to 0. The four control bits of the MMU Master Control register are described below.

Page Fault Identifier (PFI). This 5-bit field latches information that indicates which page descriptor register was being accessed when the access violation was detected.

System Mode Program/Data Separation Enable (SPD). While this bit is set to 1, instruction fetches and data accesses via the PC Relative addressing mode use the system mode page descriptor registers 8-15, and data references that do not use the PC Relative addressing mode use the system mode page descriptor registers 0-7. While this bit is cleared to 0, system mode page descriptor registers 0-15 are used to translate instruction and data references.

System Mode Translate Enable (STE). While this bit is set to 1, logical addresses generated in the system mode of operation are translated. While this bit is cleared to 0, addresses are passed through the MMU extended with zeros in the most significant bits and no attribute checking or modified bit setting is performed.

User Mode Program/Data Space Separation Enable (UPD). While this bit is set to 1, instruction fetches and data accesses via the PC Relative addressing mode use user mode page descriptor registers 8-15, and data references that do not use the PC Relative addressing mode use user mode page descriptor registers 0-7. While this bit is cleared to 0, user mode page descriptor registers 0-15 are used to translate instruction and data references.

User Mode Translated Enable (UTE). While this bit is set to 1, logical addresses generated in the user mode of operation are translated. While this bit is cleared to 0, addresses are passed through the MMU extended with zeros in the most significant bits and no attribute checking or modified bit setting is performed.

Page Descriptor Register Pointer. Moves of data into and out of the MMU page descriptor registers use the Page Descriptor Register Pointer. This 8-bit register contains the address of one of the page descriptor registers. When a word I/O instruction accesses I/O address FFxxF5 (descriptor select port), this register is used to access a page descriptor register. When a word I/O instruction accesses I/O address FFxxF4 (block move I/O port), this register is also used to access a page descriptor register, but after the access, this register is automatically incremented by one.

Descriptor Select Port. Moves of one word of data into and out of a page descriptor register are accomplished by writing and reading words to or from this dedicated I/O port at location FFxxF5. Any word I/O instruction can be used to access a page descriptor register via this port, provided that the Page Descriptor Register Pointer is properly initialized.

Block Move I/O Port. Block moves of data into and out of the page descriptor registers are accomplished by writing and reading words to or from this dedicated I/O port at location FFxxF4. Any word I/O instruction can be used to access page descriptor registers via this port, provided that the Page Descriptor Register Pointer is properly initialized.

Invalidation I/O Port. Valid bits can be cleared (i.e., the page descriptor registers invalidated) by writing to this dedicated 8-bit port (Table 3). Individual Valid bits can subsequently be set by software writing to the page descriptor registers. Reading this I/O port returns unpredictable data.

Table 3. Invalidation Port Table

Encoding	Registers Invalid
01_H	System Page Descriptor Registers 0-7
02_H	System Page Descriptor Registers 8-15
03_H	System Page Descriptor Registers 0-15
04_H	User Page Descriptor Registers 0-7
05_H	User Page Descriptor Registers 8-15
08_H	User Page Descriptor Registers 8-15
$0C_H$	User Page Descriptor Registers 0-15

Translation Mechanism

Address Translation. Address translation is illustrated in Figure 20. While the Program/Data Space Separation bit is cleared to 0, the 16-bit logical address is divided into two fields, a 4-bit index field used to select one of 16 page descriptor registers, and a 12-bit offset field that forms the lower 12 bits of the physical address. The physical address is composed of the 12-bit page frame address supplied by the selected page descriptor register and the 12-bit offset supplied by the logical address.

While the Program/Data Space Separation bit is set to 1, the logical address is divided into a 3-bit index field and a 13-bit offset field. The page descriptor register consists of an 11-bit Page Frame Address field. The physical address is a result of concatenating the page frame address and the logical offset. The page descriptor register is chosen by a 4-bit index field, which consists of a Program/Data Address bit from the CPU and the three Index bits from the logical address.

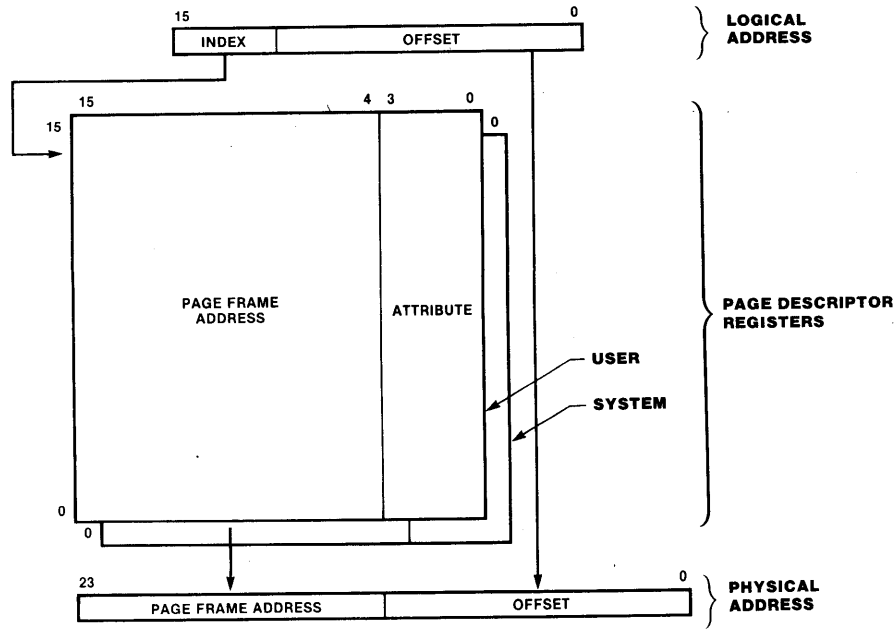


Figure 20. Address Translation

ON-CHIP MEMORY

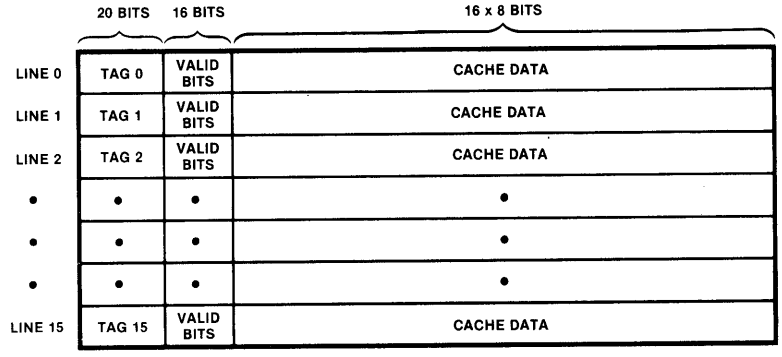
Features

- 256-byte local memory
- Configurable as high-speed cache
- Programmable to cache instructions, data or both
- Permits faster execution by minimizing external bus accesses
- Operation is transparent to user
- Configurable as local RAM with user-definable addresses

The Z800 MPU has 256 bytes of on-chip memory, which can be dedicated to memory locations programmed by the system or used as a cache for instructions or data. Its mode of use (dedicated memory or a cache) is programmable; on reset it is automatically enabled for use as a cache for instructions only.

On-Chip Memory Architecture

The on-chip memory is organized as 16 lines of 16 bytes each. Each line can hold a copy of 16 consecutive bytes in physical memory locations whose 20 most significant bits of physical address are identical. Each byte in the cache has an associated Valid bit that indicates whether the cache holds a valid copy of the memory contents at the associated physical memory location. Figure 21 illustrates the cache organization.



Tag n = the 20 Address bits associated with line n
Valid bits = 16 bits that indicate which bytes in the cache contain valid data
Cache data = 16 bytes

Figure 21. Cache Organization

2259-020, 021

The on-chip memory has two modes of operation. If the Memory/Cache bit in the Cache Control register is set to 1, then the 256 bytes of on-chip memory are treated as physical memory locations. Memory accesses to addresses covered by the on-chip memory do not generate bus transactions on the external bus and hence the accesses are faster. In this mode, the valid bits are ignored.

If the Memory/Cache bit is cleared to 0, then the 256 bytes of on-chip memory are treated as a cache memory. The lines are allocated using a least-recently used (LRU) algorithm. When a cache "miss" on a read occurs (and the MMU does not assert cache inhibit), the line in the cache that has been least recently accessed is selected to hold the newly read data. All bytes in the selected line are marked invalid except for the bytes containing the newly accessed data. On a cache miss, one or two bytes, depending on the bus size, are fetched from main memory. Except for burst mode instruction fetches, the cache does not pre-fetch beyond the currently-requested address. A cache miss on a data write does not cause a line to be allocated to the memory location accessed.

The cache can hold both instructions and data. Two control bits in the Cache Control register can be separately set to enable the cache to hold instructions and to hold data. If the cache contains data, writes to data at locations contained in the cache also cause external bus transactions to update the appropriate memory location.

Both the CPU and the on-chip DMAs access the cache. For the CPU, if the MMU is enabled, the access can be either cacheable or non-cacheable, depending on the value of the Cacheable bit in the page descriptor register used to translate the logical address. If the MMU is not enabled, all memory transactions are considered to be cacheable. Two bits in the Cache Control register, the Cache Instructions Disable bit and the Cache Data Disable bit, further determine the operation of the cache for various situations. These bits enable the cache for instructions and for data.

When the on-chip memory is used as fixed memory locations, neither the Cache Instruction Disable or Cache Data Disable bits are used, and no distinction is made as to whether the CPU is accessing data or instructions.

In general, when devices such as on-chip DMAs transfer data to the memory, the cache data is modified if the write is to a valid location in the cache but the LRU mechanism is unaffected. Also, for the EPU to memory transfer, if the cache contains valid locations that are updated by an EPU transaction, the on-chip cache is also updated.

Cache Control Register. The operation of the on-chip memory is controlled by an 8-bit Cache Control register (Figure 22) that is accessed using a load control instruction. This register contains five control bits; all other bits must be cleared to 0.

Figure 22. Cache Control Register

The bits in this register are:

High Memory Burst Capability (HMB). This 1-bit field specifies whether a memory burst transaction occurs when the MMU is enabled and there is a 1 in bit 15 of the selected page descriptor register (0 = burst mode not supported, 1 = burst mode supported).

Low Memory Burst Capability (LMB). This 1-bit field specifies whether a memory burst transaction occurs when the MMU is disabled or when the MMU is enabled and there is a zero in bit 15 of the selected page descriptor register (0 = burst mode not supported, 1 = burst mode supported).

Cache Data Disable (D). While this bit is cleared to 0, data fetches are copied into the cache if the M/C bit = 0 (cache mode). If M/C = 1, the state of this bit is ignored.

Cache Instructions Disable (I). While this bit is cleared to 0, instruction fetches are copied into the cache when the M/C bit = 0 (cache mode). When M/C = 1, the state of this bit is ignored.

Memory/Cache (M/C). While this bit is set to 1, the on-chip memory is to be accessed as physical memory; while it is cleared to 0, the memory is accessed associatively as a cache.

If the on-chip memory is to be used as fixed memory locations, the user can programmably select the ranges of memory addresses for which the on-chip memory responds.

CLOCK OSCILLATOR

The Z800 MPU has an on-chip clock oscillator/generator that can be connected to a crystal or any suitable clock source. The bus timing clock generated from the on-chip oscillator is output for use by the rest of the system. The frequency of the processor clock is one-half that of the fundamental frequency of the crystal.

REFRESH

The Z800 MPU has an internal mechanism for refreshing dynamic memory. This mechanism can be activated by setting the Refresh Enable bit in the Refresh Rate register to 1. Memory refresh is performed periodically at a rate specified by the Refresh Rate register. Refresh transactions are identical to memory transactions except that different status signals are used and no data is transferred. They can be inserted immediately after the last clock cycle of any bus transaction, including an internal operation.

While the Refresh Enable bit is set to 1, the value of the 6-bit Rate field in the Refresh Rate register determines the time between successive refreshes (the refresh period). When Rate = 0, the refresh period is 256 processor clock cycles; when Rate = n (n > 0) the refresh period is 4n. The Rate and Refresh Enable control bits are programmed via an I/O instruction.

The refresh transaction is generated as soon as possible after the refresh period has elapsed (generally after the last clock cycle of the current bus transaction). If the CPU receives an interrupt request, the refresh operation is performed first. When the Z800 CPU does not have control of the bus or is in the wait state, internal circuitry records the number of refresh periods that have elapsed and refresh cycles cannot be generated. When the CPU regains control of the bus or the Wait input signal is deactivated and the bus transaction completes, the refresh mechanism immediately issues the skipped refresh cycles. The internal circuitry can record up to 256 such skipped refresh operations.

A 10-bit refresh address is generated for each refresh operation with the refresh address being incremented by two between refreshes for 16-bit bus versions, and by one for 8-bit bus versions.

On reset, refresh is enabled, the rate is 32 processor clock cycles, and the refresh address is not affected.

The Refresh mechanism is controlled by an 8-bit control register, described below.

Refresh Rate Register. This 8-bit register (Figure 23) enables the refresh mechanism and specifies the frequency of refresh transactions.

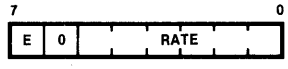


Figure 23. Refresh Rate Register

The fields in this register are:

Refresh (Rate). This field indicates in processor clock cycles the rate at which refresh transactions are to be generated; a value of n in this field indicates a refresh period of 4n, with Rate = 0 indicating 256 clock cycles.

Refresh Enable (E). When this 1-bit field is set to 1, the refresh mechanism is enabled.

UART

The Z800 UART transmits and receives serial data using any common asynchronous data-communication protocol.

Transmission and reception can be performed independently with five, six, seven, or eight bits per character, plus optional even or odd parity. The transmitter can supply one or two stop bits and can provide a break output at any time. Reception is protected from spikes by a "transient spike-rejection" mechanism that checks the signal one-half a bit time after a Low level is detected on the receiver data input; if the Low does not persist—as in the case of a transient—the character assembly process is not started. Framing errors and overruns are detected and buffered with the partial character on which they occur. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The UART uses the same clock frequency for both the transmitter and the receiver. The input for the UART clocking circuitry is derived from counter/timer 0, either from its external input line for an external clock or from the counter/timer output for a bit rate generated from the internal processor clock. The UART input clock is further scaled by 1, 16, 32, or 64 for clocking the transmitter and receiver.

Two of the DMA channels can be used independently to move characters between memory and the transmitter or receiver without CPU intervention. Both the transmitter and receiver can interrupt the CPU for processor assistance.

The UART uses two external pins, Transmit and Receive. Data that is to be transmitted is placed serially on the Transmit pin and data that is to be received is read in from the Receive pin.

Asynchronous Transmission

The Transmitter Data Output line is held marking (High) when the transmitter has no data to send. Under program control, the Send Break command can be issued to hold the Data Output line Low (spacing) until the command is cleared.

The UART automatically adds the start bit, the programmed parity bit (odd, even, or no parity), and the programmed number of stop bits to the data character to be transmitted. When the character is five, six, or seven bits, the unused most significant bits in the Transmitter Data register are automatically ignored by the UART.

Serial data is shifted from the transmitter at a rate equal to 1, 1/16th, 1/32nd or 1/64th of the clock rate supplied to the transmitter clock input (as determined by the clock scale field in the UART Configuration register). Serial data is shifted out on the falling edge of the clock input.

Asynchronous Reception

An asynchronous receive operation begins when the Receive Enable bit in the Receiver Control/Status register is set to 1. A Low (spacing) condition on the Receive input line indicates a start bit. If this Low persists for at least one-half of a bit time, the start bit is assumed to be valid and the data input is then sampled at mid-bit time until the entire character is assembled. This method of detecting a start bit improves error rejection when noise spikes exist on an otherwise marking line. If the ×1 clock mode is selected, bit synchronization must be accomplished externally; received data is sampled on the rising edge of the clock.

A received character can be read from the 8-bit Receiver Data register. The receiver inserts 1s when a character length of other than eight bits is used. If parity is enabled, the parity bit is not stripped from the assembled character for character lengths other than eight bits. For lengths other than eight bits, the receiver assembles a character length of the required number of data bits, plus a parity bit and 1s for any unused bits.

Since the receiver is buffered by one 8-bit register in addition to the receiver shift register, the CPU has enough time to service an interrupt and to accept the data character assembled by the UART. The receiver also has a buffer that stores error flags for each data character in the receive buffer. These error flags are loaded at the same time as the data character.

After a character is received, it is checked for the following error conditions:

- Parity Error: when the parity bit of the character does not match the programmed parity.
- Framing Error: if the character is assembled without any stop bits (i.e., a Low level is detected for a stop bit).
- Receiver Overrun Error: if the CPU fails to read a data character when more than one character has been received.

The Parity Error, Framing Error, and Receiver Overrun Error cause an interrupt request if the interrupt request capability is enabled. Since the Parity Error and Receiver Overrun Error flags are latched, the error status that is read reflects an error in the current character in the Receiver Data register plus any Parity or Overrun Errors detected since the last write to the Receiver Control/Status register. To keep correspondence between the state of the error buffers and the contents of the receiver data buffers, the Receiver Control/Status register must be read before the data.

Polled Operation

In a polled environment, the Receive Character Available bit in the Receiver Control/Status register must be monitored so the CPU can know when to read a character. This bit is automatically cleared when the Receiver Data register is read. To prevent overwriting data in polled operations, the transmitter buffer status must be checked before writing into the transmitter. The Transmit Buffer Empty bit in the Transmitter Control/Status register is set to 1 whenever the transmit buffer is empty.

UART Control and Status Registers

The UART operation is controlled by three control and status registers. The UART configuration register specifies the character size and parity, the clock source and scaling and loop-back enable. Both the transmitter and the receiver have their own control/status register.

UART Configuration Register. This 8-bit register (Figure 24) contains control information for both the transmitter and receiver.

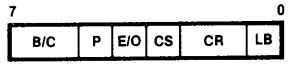


Figure 24. UART Configuration Register

The control fields for this register are:

Loop Back Enable (LB). The UART is capable of local loopback. In this mode the internal transmit data line is tied to the internal receiver line and the external receiver input is ignored. If this bit is set to 1, loop mode is enabled.

Clock Rate (CR). These two bits specify the multiplier between the clock and data rates (00 = data rate × 1, 01 = data rate × 16, 10 = data rate × 32, 11 = data rate × 64). The same rate is used for both the receiver and transmitter. If the ×1 clock rate is selected, bit synchronization must be accomplished externally.

Clock Select (CS). This bit specifies the clock input for the UART. If the bit is set to 1, the counter/timer 0 output pulse is used for bit-rate generation; if the bit is cleared to 0, the input line to counter/timer 0 provides the clock from an external source.

Parity Even/Odd (E/O). If parity is specified, this bit determines whether it is sent and checked as even or odd (1 = even).

Parity (P). If this bit is set to 1, an additional bit position (in addition to those specified in the bits/character control field) is added to transmitted data and is expected in received data. In the Receiver, the parity bit received is transferred to the CPU as a part of the character, unless eight bits/character is selected.

Bits/Character (B/C). Together, these two bits determine the number of bits to form a character. If these bits are changed during the time that a character is being assembled, the results are unpredictable (00 = 5 bits/character, 01 = 6 bits/character, 10 = 7 bits/character, 11 = 8 bits/character).

Transmitter Control/Status Register. This 8-bit register (Figure 25) specifies the operation of the transmitter.

Figure 25. Transmitter Control/Status Register

The control bits for this register are:

Transmitter Buffer Empty (BE). This bit is automatically set to 1 whenever the transmitter buffer becomes empty, and cleared to 0 when a character is loaded into the transmit buffer. This bit is in the set condition after a reset. This bit is controlled by the UART control circuitry, it can be read by an I/O read but cannot be set to 1 or cleared to 0 by an I/O write.

Value (VAL). This bit determines the value of the bits transmitted while the FRC bit is 1 and dummy characters are loaded into the transmitter buffer. When this bit is 1, a mark character (all 1s) is sent; when this bit is 0, a break character (all 0s) is sent.

Force Character (FRC). When this bit is set to 1, any character loaded into the transmitter buffer causes the transmitter output to be held High or Low (as indicated by the VAL bit) for the length of time required to transmit a char)cter; the character itself is not sent until after the current character is transmitted. This allows a program to generate a marking signal or a break of multiple-character duration simply by setting this bit to 1, setting the VAL bit to 1 or 0, and loading the appropriate number of dummy characters into the transmitter buffer.

Send Break (BRK). When set to 1, this bit immediately forces the transmitter output to the spacing condition, regardless of any data being transmitted. When this bit is cleared to 0, the transmitter returns to marking.

Stop Bits (SB). This bit determines the number of stop bits added to each asynchronous character sent. The receiver always checks for one stop bit. If this bit is set to 1, two stop bits are automatically appended to the character (and parity) sent; if this bit is cleared to 0, only one stop bit is appended.

Transmitter Interrupt Enable (IE). When this bit is set to 1, interrupt requests are generated whenever the transmitter buffer becomes empty; when this bit is cleared to 0, no requests are made.

Transmitter Enable (EN). While this bit is cleared to 0, data is not transmitted and the transmitter output is held marking. Data characters in the process of being transmitted are completely sent if this bit is cleared to 0 after transmission has started.

Receiver Control/Status Register. This 8-bit register (Figure 26) specifies the operation of the receiver. The control bits are described below.

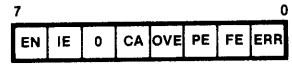


Figure 26. Receiver Control/Status Register

Receiver Error (ERR). This bit is the logical OR of the PE, OVE, and FE bits.

Framing Error (FE). This bit is automatically set to 1 for the received character in which the framing error occurred. Detection of a framing error adds an additional one-half of a bit time to the character time so the framing error is not interpreted as a new start bit. The bit is latched, so once an error occurs it remains set until the bit is cleared by software writing to this register.

Parity Error (PE). When parity is enabled, this bit is automatically set to 1 for those characters whose parity does not match the programmed sense (even/odd). This bit is latched, so once an error occurs, it remains set until it is cleared by software writing to this register.

Receiver Overrun Error (OVE). This bit is automatically set to 1 to indicate that more than two characters have been received without a read from the CPU (or DMA). Only the most recently received character is flagged with this error, but when this character is read, the error condition is latched until cleared by software writing to this register.

Receiver Character Available (CA). This bit is automatically set to 1 when at least one character is available in the receive buffer; it is automatically cleared to 0 when the Receiver Data register is read. This bit is controlled by the UART control circuitry; it can be read by an I/O read but cannot be set or cleared by an I/O write.

Receiver Interrupt Enable (IE). While this bit is set to 1, interrupt requests are generated whenever the receiver detects an error or the receiver has a character available.

Receiver Enable (EN). When this bit is set to 1, receiver operations begin. This bit should be set only after the parameters in the UART Configuration register are set.

UART Bootstrapping Option

The Z800 CPU supports an automatic initialization of memory via the UART after a reset operation. This system bootstrapping capability permits ROMless system configurations: the memory can be initialized by a serial link before the Z800 CPU fetches information from memory after the reset.

On the rising edge of reset, the AD lines are sensed; if AD_6 is being driven High, the Z800 CPU automatically enters a Halt state. The UART is also automatically initialized to receive 8-bit character data with odd parity at a $\times 16$ clock rate. An external clock source is assumed. A minimum of 15 processor clock cycles must elapse before the transmission can begin.

During the bootstrapping operation, DMA Channel 0 is used to transfer received characters into the memory. This channel is initialized as follows:

Transfer Descriptor register—IE, EPS, and TC cleared, ST-byte transfer, BRP-continuous, TYPE-flowthrough, DAD-Auto-increment memory address

DMA Master Control register—DOR and EOP set

Count register—0100 (256 bytes to be transferred)

Destination Address register—000000 (starting memory address = 0)

Source Address register—undefined (not used when DMAO is linked to UART)

Characters received are placed in memory starting at physical memory location zero. If an error occurs, the Z800 CPU drives the Transmitter Output line Low. External circuitry monitoring this line can use this signal to cause the transmitting device to begin the initialization procedure again, starting with a reset and AD_6 asserted on the rising edge of $\overline{RESET}$.

After 256 bytes of data have been transferred, the Z800 CPU automatically begins execution by fetching the first instruction from memory location 0.

DMA CHANNELS

The Z800 MPU has four on-chip Direct Memory Access (DMA) channels to provide high bandwidth data transmission capabilities. There are two types of DMA channels; two support flyby transactions and the other two do not. The two types of DMA channels otherwise have identical capabilities, although they have different priorities with respect to interrupt requests and bus requests.

Each DMA channel is a powerful and versatile device for controlling and processing transfers of data. Its basic function of managing CPU-independent transfers between two ports is augmented by an array of features requiring little or no external logic in systems using an 8- or 16-bit data bus.

Transfers can be performed between any two ports (source and destination), including memory-to-I/O, I/O-to-memory, memory-to-memory, and I/O-to-I/O. Except for flyby, two port addresses are automatically generated for each transaction and can be either fixed or incrementing/decrementing.

During a transfer, a DMA channel assumes control of the system address and data bus. Data is read from one addressable port and written to the other addressable port, byte-by-byte or word-by-word. The ports can be programmed to be either system main memory or peripheral I/O devices.

For both flyby and flowthrough DMA transactions, if the destination is a memory location that corresponds to an entry in the on-chip memory (either cache or fixed memory location), the on-chip memory is updated to reflect the new contents of the memory location.

Except in flyby mode, two 24-bit addresses are generated by the DMA for every transfer operation, one address for the source port and another for the destination port. Two readable address counters (three bytes each) keep the current address of each port.

The DMA devices use the same memory and I/O timing as the CPU for bus transactions, as indicated by the appropriate bus timing register.

Modes of Transfer Operation

Each DMA can be programmed to operate in one of three transfer modes:

- *Single Transaction.* Data operations are performed one byte or word at a time.
- *Burst.* Data operations continue until a port's Ready line to the DMA goes inactive.
- *Continuous.* Data operations continue until the end of the programmed block of data is reached or if an end of process has been signaled before the system bus is released.

In all modes, once a byte or word of data is read by the DMA channel, the operation is completed in an orderly fashion, regardless of the state of other signals (including a port's Ready line).

Pin Descriptions

Each DMA channel has a Ready input line. In addition, two DMA channels have a flyby output line to support high speed data transfers between I/O devices and memory.

The flyby output is asserted by the DMA channel to signal a peripheral device associated with the DMA channel that it should participate in the data transmission during the current flyby bus transaction.

The Ready line is sampled on the rising edge of each processor clock cycle. If Ready is active, the DMA channel requests control of the external system bus to perform the DMA transaction. When the external system bus is available for DMA transfers, the DMA channel with a request pending and the highest priority assumes bus mastership. The priority of DMA channels from highest to lowest is: DMA0, DMA1, DMA2, and DMA3. A DMA channel in burst mode relinquishes bus mastership to a higher priority DMA channel only when its Ready line is deasserted (or EOP is signaled or terminal count is reached). A DMA channel in continuous mode relinquishes bus mastership only when EOP is signaled or terminal count is reached.

Priority of On-Chip DMA Channels and External Bus Requesters

The on-chip DMA channels are arranged in a daisy chain with the external Bus Request input line being the "next lower bus requester" on this chain. The on-chip DMAs behave as if they were external bus requestors with respect to acquiring the bus, relinquishing the bus, and priority access to the bus.

End-of-Process

If the end-of-process (EOP) capability is enabled, transfers by DMA channels can be prematurely terminated by a Low on Interrupt A line during the transfer. This capability is programmed by a control bit in the DMA Master Control register. EOP occurs regardless of the setting of the Interrupt A Enable bit in the Master Status register. When an EOP is signaled, the EOP Signaled (EPS) bit in the Transaction Descriptor register of the active DMA channel is set to 1 and the Enable bit is cleared to 0. If interrupt requests are enabled (IE = 1 in the Transaction Descriptor register), an interrupt request is generated by the channel that was active when the EOP was signaled. After an EOP has been signaled, the DMA relinquishes the bus within 16 cycles of the last DMA bus transaction.

If the End-Of-Process signal on Interrupt A line is still asserted when the CPU is bus master, the signal is interpreted as an interrupt request; thus both the DMA channel and the external EOP generating device can request interrupts simultaneously. Separate mask bits in the Master Status register enable the CPU to accept interrupts from these two sources.

On a flowthrough transaction, if the EOP signal is received while the information is being read into the Z800 MPU, the transfer is aborted and the data is not written out from the Z800 MPU.

DMA Linking

The DMA devices can be linked together to achieve DMA transfers to non-contiguous memory locations (linked operation). Bits in the DMA Master Control register allow DMA3 to be linked to DMA1 and DMA2 to be linked to DMA0, when DMA0 and DMA1 have flyby capabilities. If the appropriate bit is set to 1 in the DMA Master Control register, the master DMA (0 or 1) signals its linked DMA each time its transfer is complete (count = 0). This acts as an internal ready input to the linked DMA that reloads the master DMA control registers.

Words are loaded into the master DMA control registers in the following order: Destination Address register (two words), Source Address register (two words), Count (one word), Transfer Descriptor register (one word). After six words have been transferred, the master DMA deasserts its internal ready line and begins the transfer of the next block of data. The linked DMA can be programmed to interrupt the CPU on "count equal 0" (to notify software that the last block is being transferred) or the master DMA can be programmed to interrupt the CPU on "count equals 0" when the last block move is programmed into the master DMA (to notify software that the entire sequence of transfers is completed). When linking is enabled, the external Ready line is not asserted by the master DMA when count equals zero; also, both master and linked DMAs generate interrupts whenever the programmed condition arises.

When programming linked DMAs, the last word to be programmed must be the master DMA's Transaction Descriptor register. Also, the linked DMA must be programmed before the master DMA's status register is programmed.

DMA Master Control Register. This 16-bit register (Figure 27) specifies the general configuration of the four on-chip DMA channels: the linking of the DMA channels, the software ready enables, edge detection enables for the Ready lines, and EOP enable.

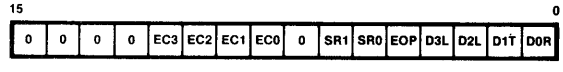


Figure 27. DMA Master Control Register

The fields in this register are:

DMA0 to Receiver Link (D0R). When this bit is set to 1, DMA channel 0 is linked to the UART receiver.

DMA1 to Transmitter Link (D1T). When this bit is set to 1, DMA channel 1 is linked to the UART transmitter.

DMA2 Link (D2L). When this bit is set to 1, DMA channel 2 is linked to DMA channel 0.

DMA3 Link (D3L). When this bit is set to 1, DMA channel 3 is linked to DMA channel 1.

End-of-Process (EOP). When this bit is set to 1, the INT_A line is used as an end-of-process signal for the active DMA channel.

Software Ready for DMA0 (SR0). When this bit is set to 1, DMA channel 0 requests service if enabled.

Software Ready for DMA1 (SR1). When this bit is set to 1, DMA channel 1 requests service if enabled.

Enable Count n (EC_n). When bit EC_n is set to 1, edge detection circuitry is enabled on Ready line n.

DMA Channel Control Registers

Transaction Descriptor Registers. These four 16-bit registers, one for each channel, (Figure 28) describe the type of DMA transfer to be performed and contain control and status information.

Figure 28. Transaction Descriptor Register

The fields in this register are:

End-of-Process Signaled (EPS). This bit is set to 1 automatically when the channel is active and an end-of-process is signaled on the Interrupt A input line, thus prematurely terminating the transfer.

Destination Address Descriptor (DAD). The setting of this 3-bit field indicates the type of location (memory or I/O) and how the address is to be manipulated (incremented, decremented or left unchanged), as shown in Table 4.

Table 4. SAD and DAD Encodings

Encoding	Address Modification Operation
0 0 0	Auto-increment memory location
0 0 1	Auto-decrement memory location
0 1 0	Memory address unmodified by transaction
0 1 1	Reserved
1 0 0	Auto-increment (by 1) I/O location
1 0 1	Auto-decrement (by 1) I/O location
1 1 0	I/O address unmodified by transaction
1 1 1	Reserved

Transfer Complete (TC). This bit is set to 1 automatically when the count register has reached zero.

Transaction Type (Type). This 2-bit field specifies flyby or flowthrough type of operation or count option (00 = flowthrough, 01 = count option, 10 = flyby write, 11 = flyby read). In flowthrough mode of operation, two bus transactions occur for each DMA operation—a read from the source followed by a write to the destination. In a flyby operation, only one bus transaction occurs for each DMA operation. In flyby write to memory, the flyby output pin is pulsed instead of an I/O transaction being performed and the contents of the Destination Address register are output to specify the memory location. In flyby read from memory, the flyby output pin is pulsed instead of an I/O transaction being performed and the contents of the Source Address register are output to specify the memory location. Only two DMAs have flyby capability. In the count option type of operation, the DMA acts as a counter and the BRP field governs the counting frequency; the "count each" option decrements the count register once for each High-to-Low transition on the Ready line while the DMA enable bit is set to 1; the gate option decrements the count register once for each eight internal processor clock cycles while the Ready line is Low and the DMA enable bit is set to 1.

Bus Request Protocol (BRP). The setting of these two bits indicates the mode of DMA operation (Table 5); their interpretation depends on whether the channel is programmed for DMA operations or with the count option.

Table 5. Bus Request Protocol (BRP)

Encoding	DMA	Counter/Timer
0 0	Single Transaction	Count each
0 1	Burst	Gated count
1 0	Continuous	Continuous gated count
1 1	Reserved	Reserved

Size of Transfer (ST). This 2-bit field specifies the size of the entity to be transferred by the DMA channel (Table 6). For word transfers to or from memory locations, the memory address must be even (least significant bit is 0). Long word (32-bit) transfers are supported only in flyby mode, with the cache disabled.

Table 6. Size of Transaction (ST)

Encoding ST1	ST0	Size of Transfer	Number to Increment/ Decrement By
0	0	Byte	1
0	1	16-bit word	2
1	0	32-bit longword	4
1	1	Reserved	

Interrupt Enable (IE). When this bit is set to 1, the DMA generates an interrupt request at end of count or end of process. When this bit is 0, no interrupt request is generated.

Source Address Descriptor (SAD). The setting of this 3-bit field indicates the type of location (memory or I/O) and how the address is to be manipulated (incremented, decremented or left unchanged), as shown in Table 4.

DMA Enable (EN). While this bit is 1, the DMA transfer is enabled.

Count Register. This 16-bit register is programmed to contain the number of DMA transfers to be performed. When the contents of the count register reach zero, further requests on the RDY input line are ignored. The DMA channel can be programmed to generate an interrupt when the count register reaches zero.

Source Address Register and Destination Address Register. These 24-bit registers contain the 24-bit physical addresses to be used during the DMA transaction. They are not translated by the MMU. In flyby mode, only one of these registers is used to supply the address for the bus transaction as indicated in the Mode field in the Transfer Descriptor register. The format for these registers is shown in Figure 29.

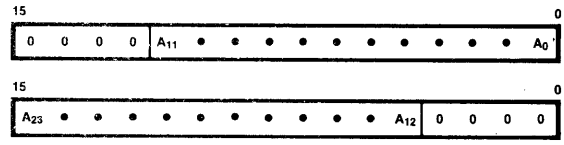


Figure 29. Source and Destination Address Registers Format

Flyby Transaction Timing

The Transaction Type field in the Transaction Descriptor register indicates whether the transaction is a read or a write. For flyby read transactions, the Source Address Descriptor indicates the transaction is a read from memory; for write flyby transactions the Destination Address Descriptor indicates the transaction is a write to memory. Additional wait states can be automatically inserted if programmed in the appropriate timing register.

COUNTER/TIMERS

The Z800 MPU's four counter/timers can be programmed by system software for a broad range of counting and timing applications. The four independently programmable channels satisfy common microcomputer system requirements for event counting, interrupt and interval timing, and general clock generation.

Three of the four counter/timers can have external input; the fourth can be used only in the timing mode.

Programming the counter/timers is straightforward: each channel is programmed with four bytes. Once started, the channel counts down, and optionally reloads its time constant automatically and resumes counting. Software timing loops are completely eliminated. Interrupt processing is simplified because each channel uses a unique vector from the Interrupt/Trap Vector Table.

Each channel is individually programmed with three registers: a configuration byte, a control byte, and a time-constant word. The configuration byte selects the operating mode (counter or timer), enables or disables the channel interrupt, and selects certain other operating parameters. In the timing mode, the CPU processor clock is divided by four for input to the counter/timers. The time-constant word contains a value from 0 to 65,535.

During operation, the individual counter channel counts down from the present time-constant value. In counter mode operation, the counter decrements on each of the input pulses until the count/time output condition is met. Each decrement is synchronized by the scaled internal processor clock. For counts greater than 65,536, two of the counters can be programmably cascaded. When the count/time output condition is reached, the downcounter is automatically reset with the time constant value, if so programmed.

The timer mode determines time intervals without additional logic or software timing loops. Time intervals are generated by dividing the internal processor clock by four and decrementing a presettable downcounter. Thus, the time interval is an integral multiple of the processor clock period, the prescaler value four, and the time constant that is preset in the downcounter. A timer is triggered by setting the software trigger control bit in the Control/Status register or by an external input.

Three channels can generate an external output when the count/time output condition is met. The output is high when the internal presettable downcounter contains all zeros.

Each channel can be programmed to generate an Interrupt Request, which occurs only if the channel has its Interrupt Enable control bit set to 1 by software programming. When the Z800 CPU accepts the interrupt request it automatically vectors through the Interrupt Vector Table.

The four channels of the Z800 MPU are fully prioritized and fit into four different slots in the Z800 internal peripheral daisy-chain interrupt structure. Channel 0 has the highest priority and Channel 3 has the lowest. The channels have separate interrupt enables and the CPU's Master Status register has individual control bits that selectively inhibit interrupts from each channel.

Modes of Operation

Three of the counter/timer channels have two basic modes of operation: as counters or as timers. As counters they monitor external input lines and record Low to High transitions on these lines. In the timer mode, the processor clock, scaled by four, is used instead of the external input line. The duration of this counting or timing can be either continuous from initial enabling (trigger operation) or only during intervals specified by signals on an input line (gate and gate/trigger operation). The count can be automatically restarted by programming the Retrigger Enable control bit in the counter/timer's Configuration register. Channel number 2 has no external inputs, and thus operates only as a timer.

Each of the four counter/timers has a software gate and trigger facility that extends the hardware capabilities of the counter/timers.

Counting Operation. While the appropriate enabling conditions are met, the counter/timer monitors its input line for Low-to-High transitions. When such a transition occurs, the Count/Time register is decremented by 1.

Timing Operation. While the appropriate enabling conditions are met, the counter/timer monitors the internal processor clock scaled by four for Low-to-High transitions. When such a transition occurs the Count/Time register is decremented by 1.

Gate Operation. A counter/timer can be programmed to count or time only when a gating condition is met. While the counter/timer is enabled and the external gate capability is selected, an external input line is monitored; only while this line is High are the counting or timing operations performed. The software gate facility filters the state of the input line; while the software gate bit in the Command and Status register is cleared to 0, the gating condition is not met regardless of the signals on the gating line. The gate facility is illustrated in Figure 30.

Trigger Operation. A counter/timer can be programmed to count or time only after a triggering condition occurs. While the counter/timer is enabled and the external trigger capability is programmed, an external input line is monitored; only after this line makes a Low-to-High transition is a counting or timing operation performed. The software trigger facility causes the triggering condition to be met regardless of the activity of this line. The trigger operation is illustrated in Figure 31.

Gate/Trigger Operation. One input line can be used for both the gating and the triggering functions. A Low-to-High transition on this line acts as a trigger and subsequent High signals on this line function as gate signals. If non-retriggerable mode is programmed, subsequent Low-to-High transactions do not cause a trigger. Gate/Trigger Operation is shown in Figure 32.

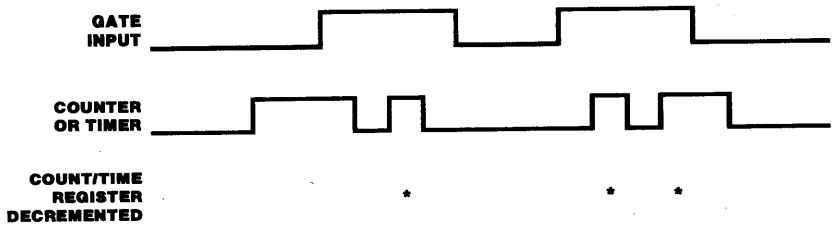


Figure 30. Gate Facility

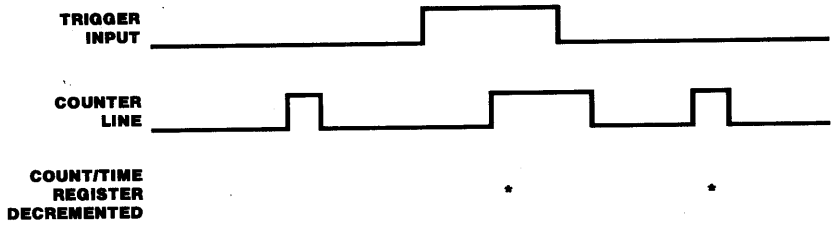


Figure 31. Trigger Operation

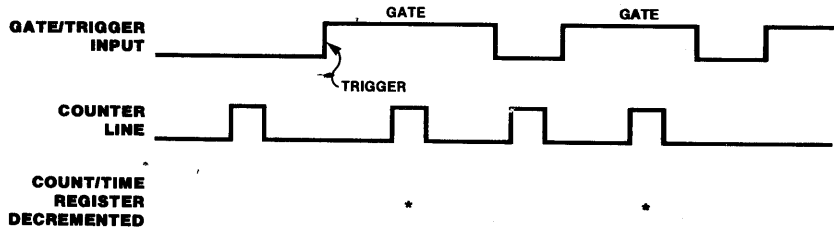


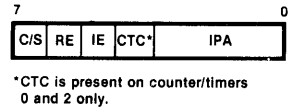
Figure 32. Gate/Trigger Operation

The software gate and trigger mechanism can also be used in this mode of operation. A software gate before a trigger (hardware or software) has no effect on the counter/timer. After a hardware or software trigger, the software gate must be set to 1 for the Count/Time register to be decremented. A software trigger after a hardware or software trigger has no effect unless the Retrigger Enable control bit is set to 1.

Counter/Timer Control and Status Registers

Each counter/timer has two 8-bit control registers and two 16-bit count registers. The Configuration register and Command and Status register determine the counter/timers's operation, the Counter/Timer Command/Status register provides information about the current operation, the Time Constant register contains the initialization value for the counter/timer, and the Count/Time register contains the current value of the count in progress.

Counter/Timer Configuration Register. This 8-bit register (Figure 33) specifies the counter/timer's mode of operation: the pin configuration, whether an interrupt request is generated, and whether the countdown sequence is automatically restarted when the count reaches zero or when a trigger occurs.



*CTC is present on counter/timers 0 and 2 only.

Figure 33. Counter/Timer Configuration Register

The fields in this register are:

Input Pin Assignments (IPA). This 4-bit field specifies the functionality of the input lines associated with the counter/timer and whether the counter/timer monitors an external input (counting operation) or uses the scaled internal processor clock (timing operation). The four bits in this field can be associated with enabling output generation (EO), selecting the external signal or internal clock (C/T), enabling the gating facility (G), and enabling the triggering facility (T). The selected options determine the functions associated with each input line associated with the counter/timer, as illustrated in Table 7.

Counter/Timer Cascade (CTC). When this bit is set to 1, counter/timers 0 and 1 and/or counter/timers 2 and 3 form a 32-bit counter. When used as 32-bit counter/timers, the control and status registers corresponding to counter/timers 0 and 2 are not used, with the exception

Table 7. Input Pin Functionality

IPA Field				Pin Functionality		
EO	C/T	G	T	Counter/Timer I/O	Counter/Timer Input	Notes
0	0	0	0	Unused	Unused	Timer
0	0	0	1	Unused	Trigger	Timer
0	0	1	0	Gate	Unused	Timer
0	0	1	1	Gate	Trigger	Timer
0	1	0	0	Unused	Input	Counter
0	1	0	1	Trigger	Input	Counter
0	1	1	0	Gate	Input	Counter
0	1	1	1	Gate/Trigger	Input	Counter
1	0	0	0	Output	Unused	Timer
1	0	0	1	Output	Trigger	Timer
1	0	1	0	Output	Gate	Timer
1	0	1	1	Output	Gate/Trigger	Timer
1	1	0	0	Output	Input	Counter
1	1	0	1	Unused	Unused	Reserved
1	1	1	0	Unused	Unused	Reserved
1	1	1	1	Unused	Unused	Reserved

of the CTC bits in the counter/timer configuration registers. The CTC bits in the counter/timer configuration registers of counter/timers 1 and 3 are never used.

Interrupt Enable (IE). While this bit is set to 1 the counter/timer generates an interrupt request when the count/time output condition is met. While this bit is 0 no interrupt request is generated.

Retrigger Enable (RE). While this bit is set to 1, the time constant value is automatically loaded into the Count/Time register when a trigger input is received while the counter/timer is counting down. While this bit is 0, no reloading occurs.

Continuous/Single Cycle (C/S). While this bit is set to 1, the countdown sequence is automatically restarted when the count reaches zero by loading the time constant value into the Count/Time register. While this bit is 0, no reloading occurs.

Counter/timer channel 2 can be programmed as a counter. However, since it has no external inputs to count, this is not a useful mode of operation.

Counter/Timer Command/Status Register. This 8-bit register (Figure 34) provides software control over the operation of the counter/timer and reflects the current status of the counter/timer's operation. Control bits in this register enable the counter/timer's operation and provide software gate and trigger capabilities. Status bits indicate whether a count is in progress, the count/time output condition has been reached, or the condition has been reached a second time.

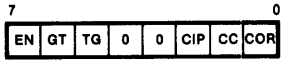


Figure 34. Counter/Timer Command/Status Register

The fields of this register are:

Count Overrun (COR). When this bit is set to 1 the count/time output condition has been reached and the CC bit is set to 1, thus indicating a count overrun condition. While this bit is cleared to 0, the count/time output condition has not been reached with the CC bit set since the time the CC bit was cleared by software. This bit can be read or written (set or cleared) by software I/O instructions.

Count/Time Output Condition has been Met (CC). When this bit is set to 1 the Count/Time register has been decremented to zero by the counter/timer control circuitry in single cycle mode, or the Count/Time register has been reloaded in continuous mode. When this bit is cleared to 0 the count has not reached the count/time output condition since the bit was cleared by software. This bit can be read or written (set or cleared) by software I/O instructions.

Count in Progress (CIP). While this bit is set to 1 the counter/timer is operating and the Count/Time register is non-zero; while this bit is cleared to 0 the counter/timer is not operating. This bit is controlled by the counter/timer control circuitry; it can be read by an I/O read but cannot be set or cleared by an I/O write instruction.

Software Trigger (TG). When this bit is set to 1 (and the trigger operation of the counter/timer is enabled), if the Enable bit is also set to 1, the trigger operation is enabled on the rising edge of the first processor clock period following the setting of this bit from a previously cleared value. That is, if a hardware trigger has not already occurred, the contents of the Time Constant register are loaded into the Count/Time register and the countdown sequence begins. If a hardware trigger has already occurred, then if Retrigger Enable is set to 1, the counter/timer is retriggered; otherwise, setting this bit has no effect. Writing a 1 in this field when the previous value was 1 has no effect on the operation of the counter/timer. When this bit is cleared to zero, this bit has no effect on the operation of the counter/timer.

Software Gate (GT). When this bit is set to 1 (and the gate operation of the counter/timer is enabled), if the Enable bit is also set to 1, operation begins on the rising edge of the first processor clock period following the setting of this bit from a previously cleared value. Writing a 1 in this field when the previous value was 1 has no effect on the operation of the counter/timer. When this bit is cleared to 0, the countdown sequence is halted.

Enable (EN). While this bit is set to 1, the counter/timer is enabled; operation begins on the rising edge of the first processor clock period following the setting of this bit from a previously cleared value. Reset clears this bit. While this bit is cleared to 0, the value in the Time Constant register is constantly transferred to the Count/Time register. If the Time Constant register is all zeros, the output of the counter/timer is one. Thus, when the counter/timer is not enabled, the counter/timer output in conjunction with the Time Constant register can be used as an I/O port. Writing a 1 in this field when the previous value was 1 has no effect on the operation of the counter/timer. While this bit is 0, the counter/timer performs no operation during the next (and subsequent) processor clock periods.

Time-Constant Register. This 16-bit register holds the value that is automatically loaded into the Count/Time register when the counter/timer is enabled, or in the continuous or retrigger mode when the count reaches zero or the trigger is asserted, respectively. This register can be read or written by I/O instructions.

Count/Time Register. This 16-bit register holds the current value of the count or timing in progress. It is automatically loaded from the Time-Constant register, and can be read by software using the I/O read instructions.

Pin Descriptions

Counter/timers 0, 1, and 3 have two external input lines associated with them. The I/O lines transfer signals between the counter/timers and external devices. The input lines receive signals from external devices for the counter/timers. The interpretations of the signals on these lines is determined by the Input Pin Assignment field in the Configuration register.

MULTIPROCESSOR MODE OF OPERATION

Features

- Allows global memory areas for shared resources
- Global memory addresses are user-specified
- Separate requests for local and global buses
- Requesting mechanism is transparent to user
- Easily interfaces to external arbiters

One mode of operation for the Z800 MPU is as an I/O Processor (IOP); while in this mode, the Z800 MPU also supports multiprocessor configurations. While operating as an IOP, the Z800 MPU is able to support both a local bus (of which the Z800 MPU is the default bus master) and a global bus (for which the Z800 MPU must request the bus and receive a bus grant signal before issuing a bus transaction.)

To accomplish this functionality, two pins previously used for the counter/timer 0 peripheral are dedicated to be global bus request and global bus grant lines; thus this feature is available only in the 64-pin devices. A register in the Z800 MPU bus interface unit is accessed for each bus transaction to determine whether the physical address must be accessed via the global or local bus.

Architecture

Pin Functionality. Two pins are used by the IOP for obtaining the global bus: the Global Request line is used to request the global bus, one which the CPU does not control by default (counter/timer 0 input/output), and the Global Acknowledge line receives an acknowledge of a global bus request (counter/timer 0 input).

Local Address Register. The bus interface unit distinguishes whether a bus transaction uses the local or global bus by comparing the four most significant bits of the physical address of memory (address bits 20 through 23) with a 4-bit Base field in the Local Address register (Figure 35). A mask field in this register specifies which bits are to be used. If all the corresponding address bits match the Base field bits (for those bit positions specified by the mask field), then the bus transaction can proceed on the local bus without requesting the global bus; if there is a mismatch in at least one specified bit position, then the global bus is be requested and the bus transaction does not proceed until the global bus acknowledge signal is asserted.

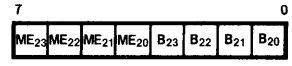


Figure 35. Local Address Register

The bits in the Local Address register are:

Base (B_n). When B_n is 1, address bit A_n must be 1 for a local bus transaction to be performed (unless Match Enable bit ME_n is 0); when bit B_n is 0, address bit A_n must be 0 for a local bus transaction to be performed.

Match Enable (ME_n). When ME_n is 1, address bit A_n is compared to base bit B_n to determine if the address requires the use of the global bus. When ME_n is 0, then any values for A_n and B_n will produce a match. If each ME_n is 0, then all bus transactions are performed on the local bus.

CPU Accesses on the Global Bus

The control of the local bus uses $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$ in the same way as in the non-multiprocessor mode of operation. The input signal $\overline{\text{BUSREQ}}$ is asynchronous to the processor clock; the CPU synchronizes $\overline{\text{BUSREQ}}$ internally. When the CPU acknowledges a local bus request by driving $\overline{\text{BUSACK}}$ active, then the CPU places all other output signals, including $\overline{\text{GREQ}}$, in 3-state. After reset the CPU acknowledges a request for the local bus before performing any transactions.

When the CPU has not granted the local bus then it can request a global bus. A timing diagram for global bus request is shown in Figure 36. First, on a rising edge of CLK, the CPU drives the address and status lines valid. AS is not asserted, however; $\overline{\text{GREQ}}$ serves the function of indicating that a valid address is on the local bus. On the next falling edge of CLK the CPU drives $\overline{\text{GREQ}}$ active. The CPU samples $\overline{\text{GACK}}$ on each falling edge of CLK until the arbiter drives $\overline{\text{GACK}}$ active and leaves $\overline{\text{BUSREQ}}$ inactive, indicating that the addressed global bus is available to tne CPU. The $\overline{\text{BUSREQ}}$ line is used by the arbiter to remove all of the devices that are simultaneously requesting the global bus, except the one device that is granted the global bus. The devices that are not granted the global bus make their $\overline{\text{GREQ}}$ inactive. The input signal $\overline{\text{GACK}}$ is asynchronous to the processor clock; the CPU synchronizes $\overline{\text{GACK}}$ internally. The CPU that was granted the bus performs one or more transactions on the global bus until the CPU no longer needs the global

bus or the CPU is prepared to acknowledge a local bus request. The CPU then drives $\overline{GREQ}$ inactive and waits for the arbiter to drive $\overline{GACK}$ inactive. The CPU relinquishes the global bus upon receipt of a local bus, DMA, or refresh request or after any transaction except for a test and set instruction (both data read and write are performed before relinquishing the bus) and for burst transfers (the entire sequence of data reads are made).

DMA Accesses on the Global Bus

Each on-chip DMA device can use the global bus to perform data transfers. The address generated during each DMA initiated transfer is compared with the contents of the Local Address register to determine whether the global bus is requested; this operation is the same as for CPU-generated requests.

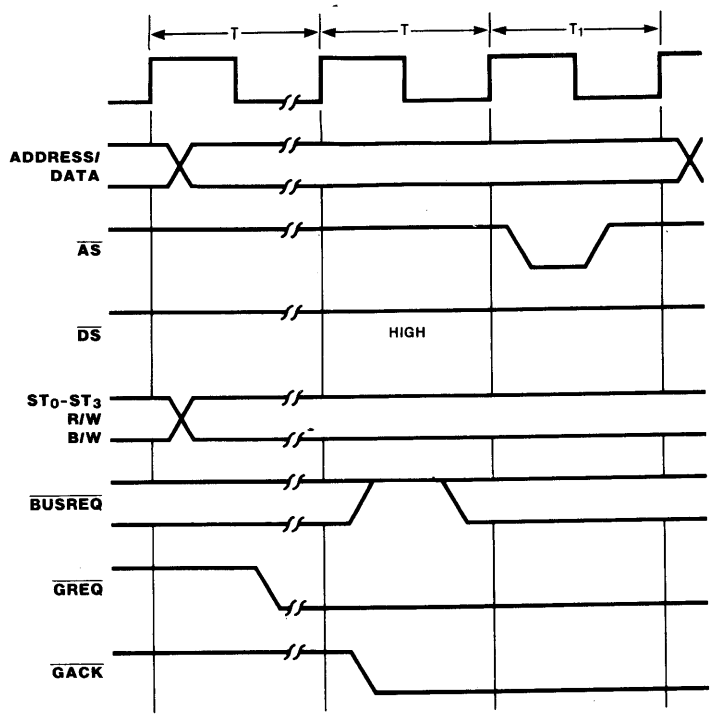


Figure 36. Multiprocessor Mode Timing

EXTERNAL INTERFACE (Z80-BUS)

Features

- 8-bit data bus
- Multiplexed address/data lines
- Supports Z80 Family peripherals

Pin Descriptions

A. *Address* (output, active High, 3-state). These address lines carry I/O addresses and memory addresses during bus transactions. Of the 16 lines, only 11 are available on the 40-pin version.

AD. *Address/Data* (bidirectional, active High, 3-state). These eight multiplexed Data and Address lines carry I/O addresses, memory addresses, and data during bus transactions.

$\overline{AS}$. *Address Strobe* (output, active Low, 3-state). The rising edge of $\overline{AS}$ indicates the beginning of a transaction and shows that the address is valid.

$\overline{BUSACK}$. *Bus Acknowledge* (output, active Low). A Low on this line indicates that the CPU has relinquished control of the bus in response to a bus request.

$\overline{BUSREQ}$. *Bus Request* (input, active Low). A Low on this line indicates that an external bus requester has obtained or is trying to obtain control of the bus.

CLK. *Clock Output* (output). The frequency of the processor timing clock is derived from the oscillator input (external oscillator) or crystal frequency (internal oscillator) by dividing the crystal or external oscillator input by two. This clock is further divided by one, two, or four (as programmed), and then output on this line.

$\overline{HALT}$. *Halt* (output, active Low, 3-state). This signal indicates that the CPU has executed a Halt instruction and is awaiting an interrupt before operation can resume.

$\overline{INT}$. *Maskable Interrupts* (input, active Low). A Low on one is available on the 40-pin version.

$\overline{\text{IORQ}}$. *Input/Output Request* (output, active Low, 3-state). This signal indicates that AD_0–AD_7 and A_{16}–A_{23} of the address bus hold a valid I/O address for a I/O read or write operation. An $\overline{\text{IORQ}}$ signal is also generated with an $\overline{\text{M1}}$ signal when an interrupt is being acknowledged, to indicate that an interrupt response vector can be placed on the data bus. Interrupt acknowledge operations occur during $\overline{\text{M1}}$ time and I/O operations never occur during $\overline{\text{M1}}$ time.

$\overline{\text{M1}}$. *Machine Cycle One* (output, active Low, 3-state). This signal indicates that the current transaction is the opcode fetch cycle of a RETI instruction execution. $\overline{\text{M1}}$ also occurs with $\overline{\text{IORQ}}$ to indicate an interrupt acknowledge cycle.

$\overline{\text{MREQ}}$. *Memory Request* (output, active Low, 3-state). This signal indicates that the address bus holds a valid address for a memory read or write operation.

$\overline{\text{NMI}}$. *Nonmaskable Interrupt* (input, falling-edge activated). A High-to-Low transition on this line requests a nonmaskable interrupt.

$\overline{\text{RD}}$. *Read* (output, active Low, 3-state). This signal indicates that the CPU or DMA peripheral is reading data from memory or an I/O device.

$\overline{\text{RESET}}$. *Reset* (input, active Low). A Low on this line resets the CPU and on-chip peripherals.

$\overline{\text{RFSH}}$. *Refresh* (output, active Low, 3-state). This signal indicates that the lower ten bits of the Address bus contain a refresh address for dynamic memories and the current $\overline{\text{MREQ}}$ signal should be used to perform a refresh read to all dynamic memories.

$\overline{\text{WAIT}}$. *Wait* (input, active Low). A Low on this line indicates that the responding device needs more time to complete a transaction.

$\overline{\text{WR}}$. *Write* (output, active Low, 3-state). This signal indicates that the bus holds valid data to be stored at the addressed memory or I/O location.

XTALI. *Clock/Crystal Input* (time-base input). Connects a series-resonant crystal or an external single phase clock to the on-chip oscillator.

XTALO. *Crystal Output* (time-base output). Connects a series-resonant crystal to the on-chip oscillator.

+5 V. *Power Supply Voltage.* (+5 nominal).

GND. *Ground.* Ground reference.

The following lines are available only on the 64-pin version:

$\overline{\text{DMASTB}}$. *DMA Flyby Strobe* (output, active Low). These lines select peripheral devices for flyby transfers.

CTIN. *Counter/Timer Input* (input, active High). These lines receive signals from external devices for the counter/timers.

CTIO. *Counter/Timer I/O* (bidirectional, active High, 3-state). These I/O lines transfer signals between the counter/timers and external devices.

$\overline{\text{IE}}$. *Input Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is toward the CPU.

$\overline{\text{OE}}$. *Output Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is away from the CPU.

$\overline{\text{PAUSE}}$. *CPU Pause* (input, active Low, Z8216 only). While this line is Low the CPU refrains from transferring data to or from on Extended Processing Unit in the system or from beginning the execution of an instruction.

RX. *UART Receive* (input, active High). This line receives serial data at standard TTL levels.

$\overline{\text{RDY}}$. *DMA Ready* (input, active Low). These lines are monitored by the DMAs to determine when a peripheral device associated with a DMA port is ready for a read or write operation. When a DMA port is enabled to operate, its Ready line indirectly controls DMA activity; the manner in which DMA activity is controlled by the line varies with the operating mode (single-transaction, burst, or continuous).

TX. *UART Transmit* (output, active High). This line transmits serial data at standard TTL levels.

Bus Operations

Two kinds of operations can occur on the system bus: transactions and requests. At any given time, one device (either the CPU or a bus requester) has control of the bus and is known as the bus master. A transaction is initiated by the bus master and is responded to by some other device on the bus. Only one transaction can proceed at a time; eight kinds of transactions can occur:

DMA Flyby. This transaction is used by the DMA peripheral to transfer data between an external peripheral and memory.

Halt. This transaction is used to indicate that the CPU is executing a halt instruction.

Internal Operation. This type of transaction does not transfer data; it indicates that the CPU is performing an operation that does not require data to be transferred on the bus.

Interrupt Acknowledge. This transaction is used by the CPU to acknowledge an interrupt and to transfer additional information from the interrupting device.

I/O. This transaction is used by the CPU or DMA peripheral to transfer data to or from an external peripheral.

Memory. This transaction is used by the CPU or DMA peripheral to transfer data to or from a memory location.

Refresh. This type of transaction performed by the refresh peripheral does not transfer data; it refreshes dynamic memory.

RETI. This transaction is generated only by the CPU and is used in conjunction with the Z8400 peripheral's interrupt logic.

Only the bus master can initiate transactions. A request, however, can be initiated by a component that does not have control of the bus. Two types of these requests can occur.

Bus. This request is used by external devices to request control of the system bus to initiate transactions.

Interrupt. This request is used to request the attention of the CPU.

When an interrupt or bus request is made, it is answered by the CPU according to its type. For an interrupt request, the CPU initiates an interrupt acknowledge transaction and for bus requests, the CPU enters bus disconnect state, relinquishes the bus, and activates an Acknowledge signal.

Finally, the Z800 MPU itself may not be the system bus master. See the multiprocessor mode section for a discussion of this capability.

Transactions

Information transfers (both instructions and data) to and from the Z800 MPU are accomplished through the use of transactions. All transactions start when $\overline{AS}$ is being driven Low and then raised High. This signal can be used to latch Z800 MPU addresses to de-multiplex the Z800 Address/Data lines required by Z80 Family peripherals. Coincident with $\overline{AS}$ assertion, the Output Enable line is also asserted.

If the transaction requires an address, it is valid on the rising edge of $\overline{AS}$. No address is required for Interrupt Acknowledge.

The Read and Write lines are used to time the actual data transfer. (Refresh transactions do not transfer any data and thus do not activate $\overline{RD}$.) For write operations, a Low on $\overline{WR}$ indicates that valid data from the bus master is on the $\overline{AD}$ lines. The Output Enable line is also activated with $\overline{WR}$. For read operations, the bus master makes the $\overline{AD}$ lines 3-state before driving $\overline{RD}$ Low so that the addressed device can put its data on the bus. The bus master samples this data on the falling clock edge just before raising $\overline{RD}$ High. The Input Enable line is also activated with $\overline{RD}$.

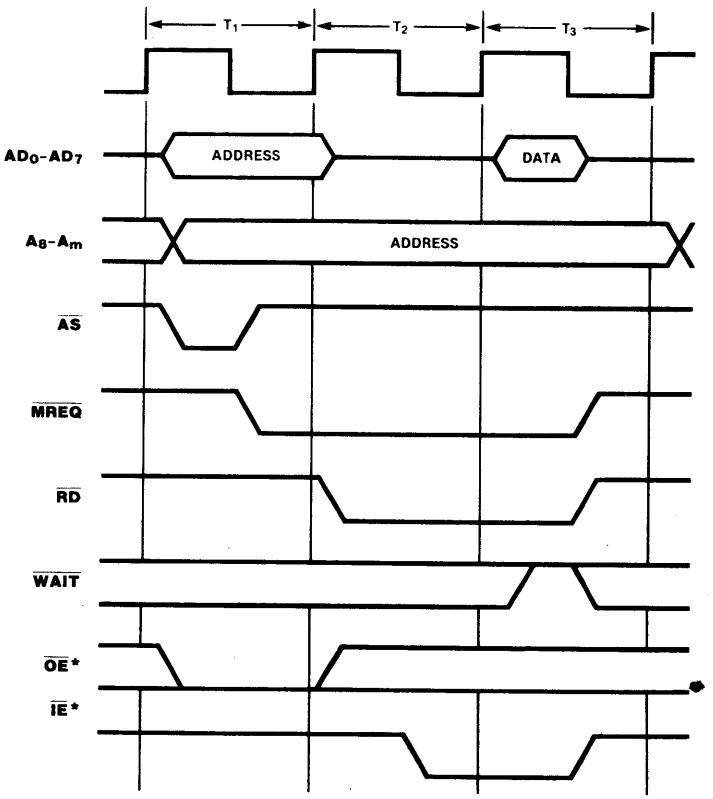
Wait. The Wait line is sampled on the falling clock edge when data is to be sampled (i.e. when $\overline{RD}$ or $\overline{WR}$ rises). If the Wait line is Low, another cycle is added to the transaction before data is sampled ($\overline{RD}$ or $\overline{WR}$ rises). In this added cycle, and all subsequent cycles added due to $\overline{WAIT}$ being Low, the Wait line is sampled on the falling edge and, if it is Low, another cycle is added to the transaction. In this way, the transaction can be extended by external devices to an arbitrary length to accommodate (for example) slow memories or I/O devices that are not yet ready for data transfer.

The $\overline{WAIT}$ input is synchronous, and must thus meet the specified setup and hold times in order for the Z800 MPU to function correctly. This requires asynchronously-generated $\overline{WAIT}$ signals to be synchronized to the CLK output before they are input into the Z800 MPU. Automatic wait states can also be generated by programming the Bus Timing and Control register and Bus Timing and Initialization register; these are inserted in the transaction before an external $\overline{WAIT}$ signal is sampled.

Memory Transactions. Memory transactions move instructions or data to or from memory when the Z800 MPU makes a memory access. Thus, they are generated during program execution to fetch instructions from memory and to fetch and store memory data. They are also generated to store old program status and fetch new program status during interrupt and trap handling, and are used by DMA peripherals to transfer information. A memory transaction is three bus cycles long unless extended with wait states, as explained previously.

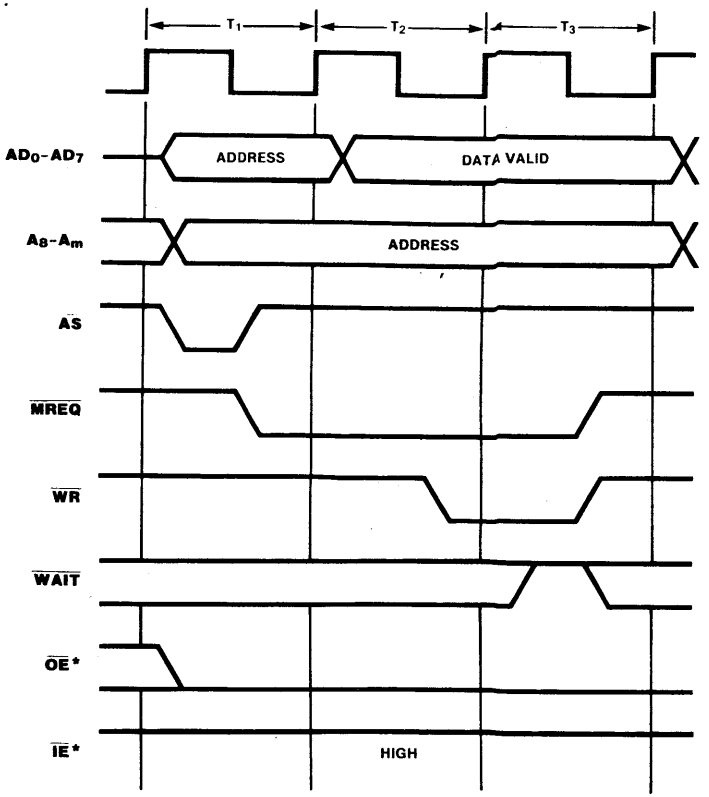
During the first bus cycle, $\overline{AS}$ is asserted to indicate the beginning of a transaction. The $\overline{MREQ}$ signal goes active during the second half of this bus cycle, which indicates a memory transaction. For a Read operation (Figure 37), $\overline{RD}$ is activated during the first half of the second bus cycle; Output Enable is deasserted at the beginning of the second cycle and Input Enable is asserted during the second half of the second cycle. The CPU samples information from the memory on the Address/Data bus with the falling edge of the clock during the third bus cycle and this same edge is used to deassert $\overline{MREQ}$, and $\overline{IE}$. Thus the data has already been sampled before $\overline{RD}$ is deasserted. For a Write operation (Figure 38) the $\overline{WR}$ line is asserted during the second half of the second cycle.

RETI Transactions. These transactions (Figure 39) are similar to two memory read transactions except that $\overline{M1}$ is asserted throughout each read transaction, falling early in the first bus cycle, and that $\overline{MREQ}$, $\overline{M1}$, $\overline{RD}$ and $\overline{IE}$ are deasserted on the rising edge of the clock following the third cycle. Each of the read transactions is followed by a minimum of three bus cycles of inactivity. These transactions are invoked when an RETI instruction is encountered in the instruction stream; they are used during the re-fetching of the instruction from memory so that interrupt logic within Z80 peripherals that monitor the bus for this instruction will function correctly.



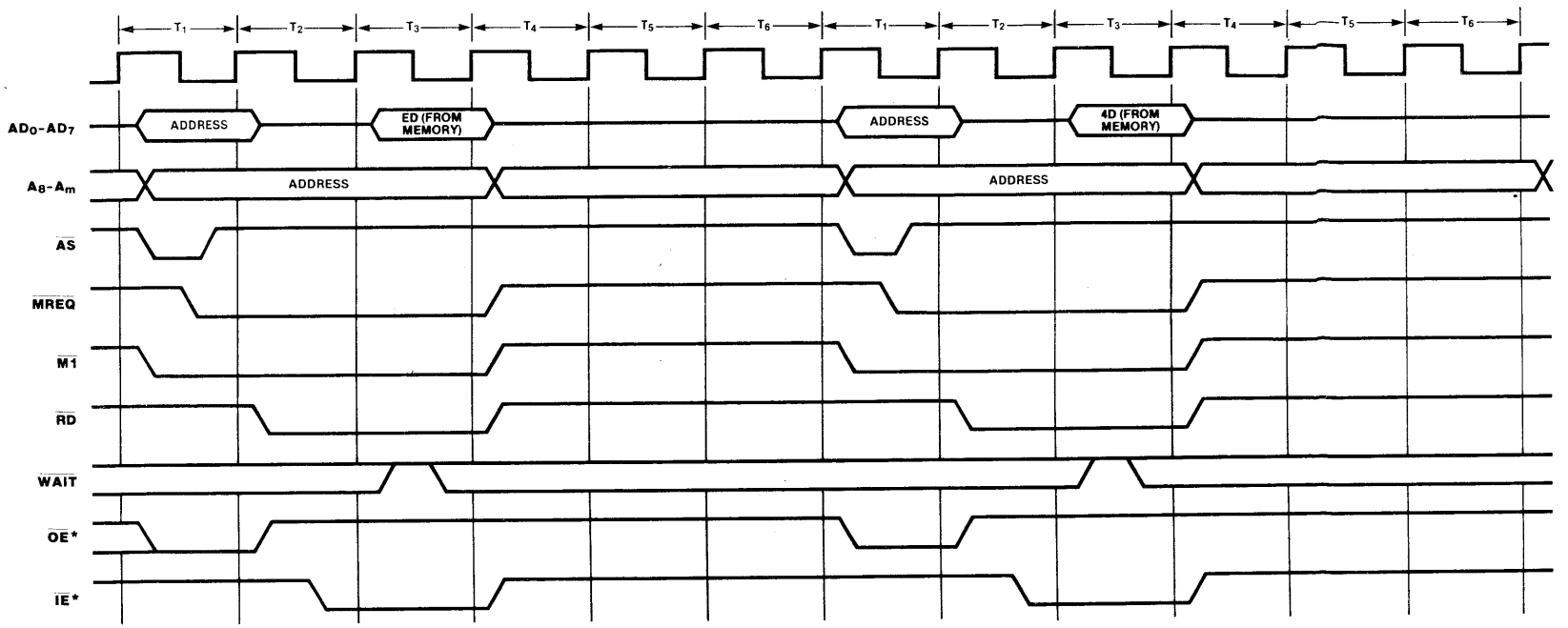
* Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 37. Memory Read Timing



* Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 38. Memory Write Timing



* Z8208 only.
8-bit Data bus only; m = 18 for Z8108, 23 for Z8208.

Figure 39. RETI Read Timing

Internal Operations and Halt Transactions. There are two kinds of bus transactions that do not transfer data: Internal Operations (Figure 40) and Halt (Figure 41). Both transactions look like a memory transaction, except that RD and WR remain High and no data is transferred. For the Internal Operation transaction, $\overline{MREQ}$ is not asserted. The Wait line is not sampled during either the Internal Operation or Halt transactions.

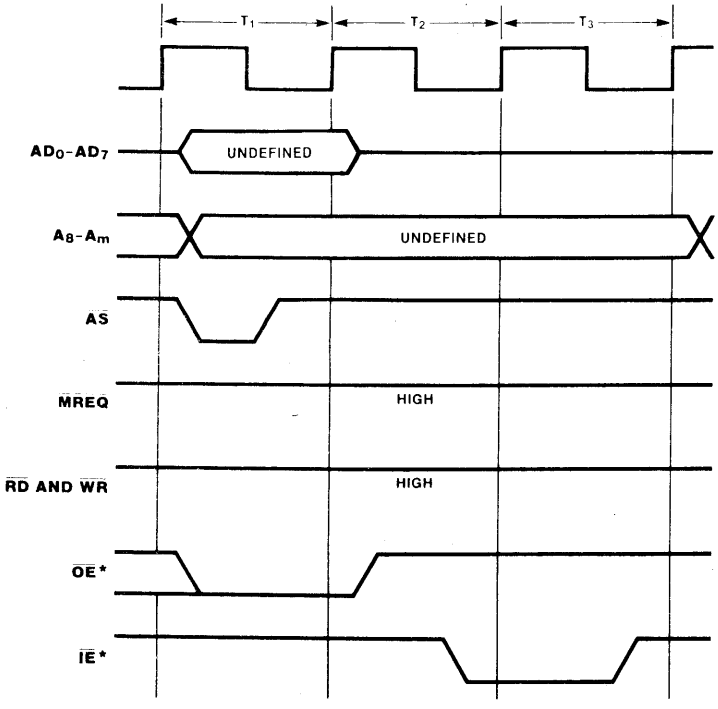
Halt transactions are identical to memory read transactions except that $\overline{HALT}$ is asserted throughout the transaction, falling during the second half of the first bus cycle, and remains asserted until an interrupt is acknowledged. This transaction is invoked when a Halt instruction is encountered in the instruction stream or a fatal sequence of traps occurs. Although the Halt transaction is three cycles, the $\overline{HALT}$ line remains asserted until an Interrupt request is acknowledged or a Reset is received. Refresh (to maintain a minimum frequency of bus transactions) or DMA transfers may occur while HALT is asserted; also, the bus can be granted. The address put out during the address phase of this cycle is the address of the Halt instruction.

I/O Transactions. I/O transactions move data to (Figure 42) or from (Figure 43) peripherals and are generated during the execution of I/O instructions.

I/O transactions are four clock cycles long at a minimum, and may be lengthened by the addition of wait cycles. The extra clock cycle allows for slower peripheral operation.

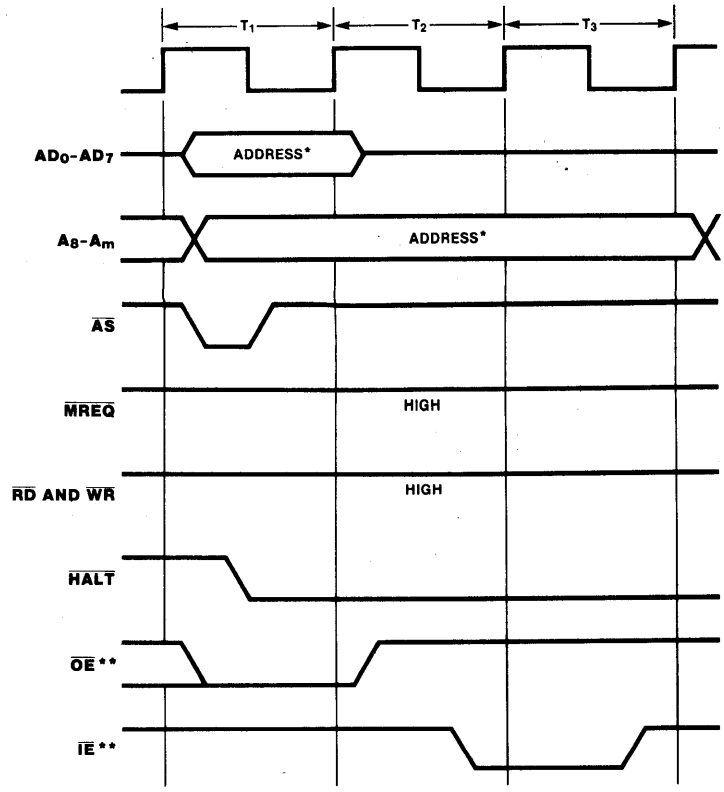
The $\overline{IORQ}$ line indicates that an I/O transaction is taking place. The I/O address is found on AD_0–AD_7 and A_8–A_{23} *when* $\overline{AS}$ *rises.*

The $\overline{IORQ}$ line and either $\overline{RD}$ and $\overline{IE}$ or $\overline{WR}$ with $\overline{OE}$ still asserted, are asserted during the second cycle. Output data to the peripheral is placed on the bus at this time; input data from the peripheral is read during the fourth cycle (unless additional wait states are inserted in the transaction).



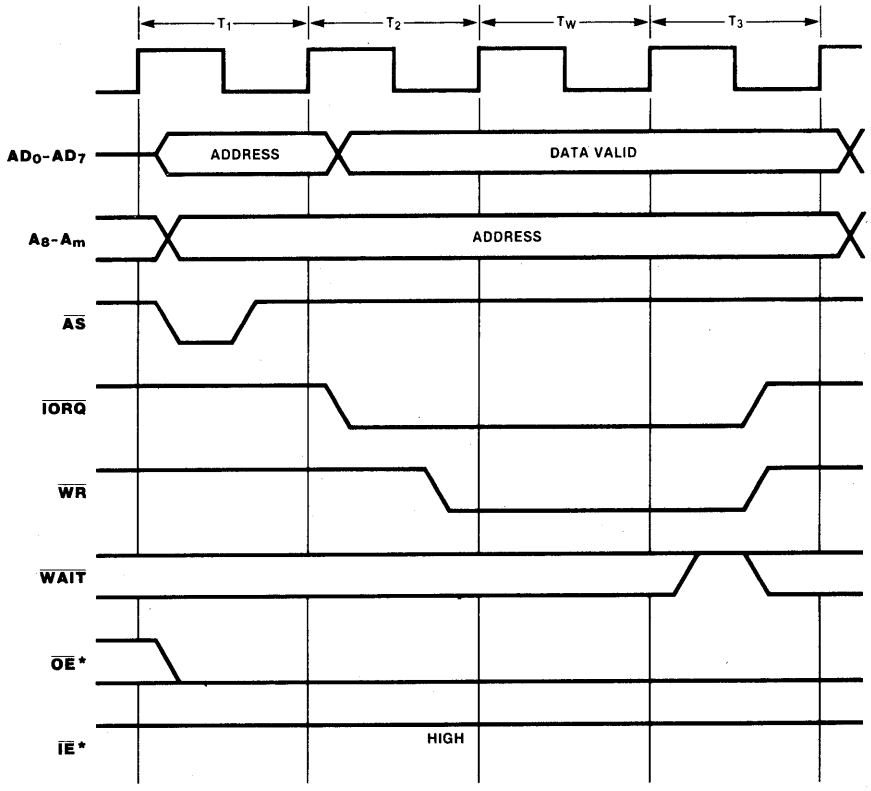
* Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 40. Internal Operation Timing



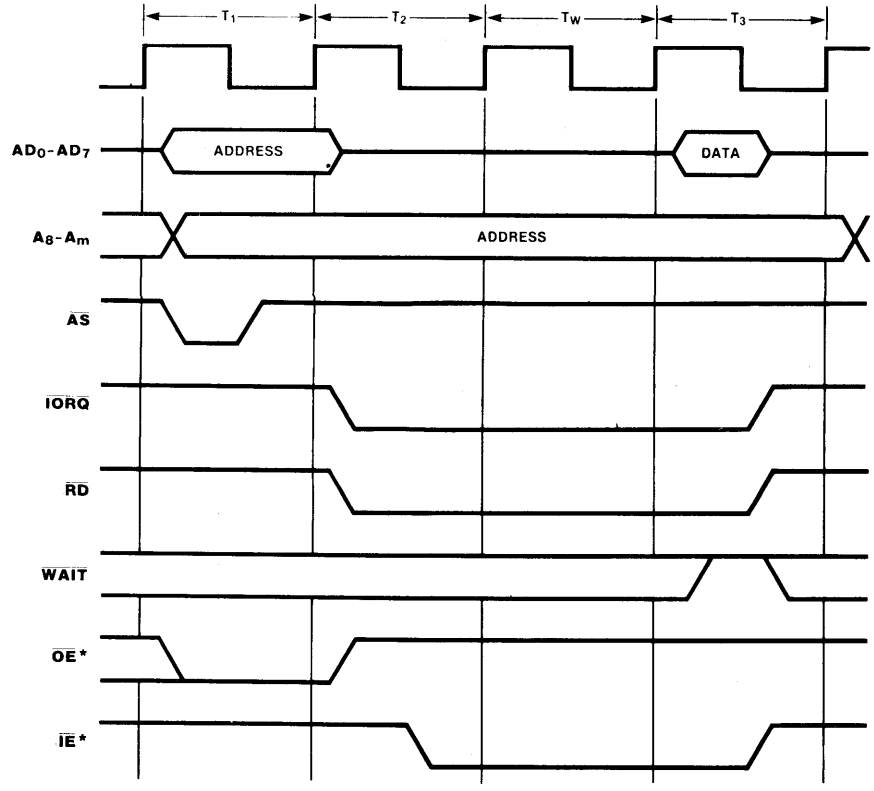
* Address of HALT instruction.
** Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 41. Halt Timing



* Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 42. I/O Write Timing



* Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 43. I/O Read Timing

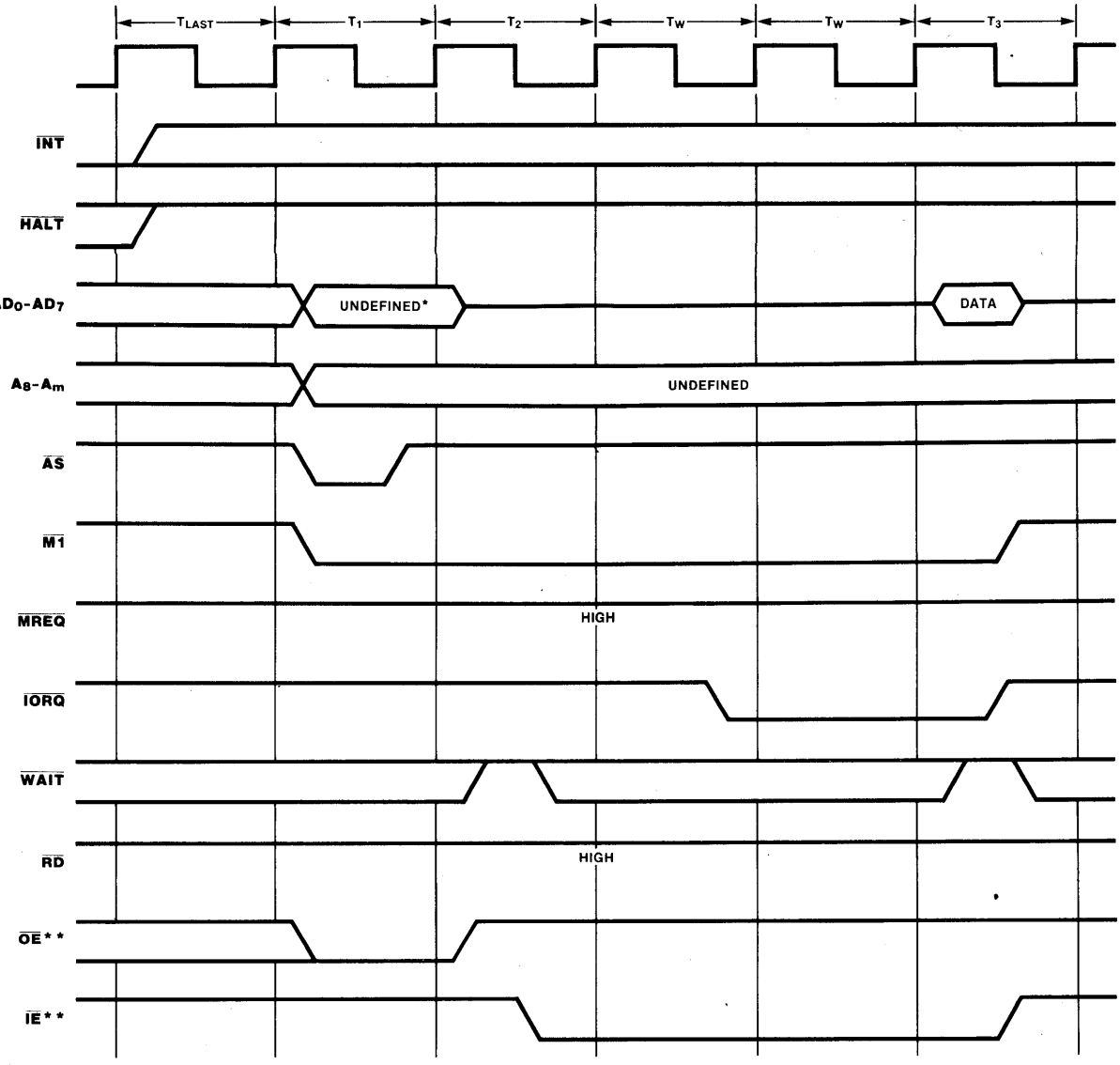
Interrupt Acknowledge Transactions. These transactions (Figure 44) acknowledge an interrupt or trap and read information from the device that generated the interrupt. The transactions are generated automatically by the hardware when an interrupt request is detected.

The Interrupt Acknowledge transactions are five cycles long at a minimum, and have two automatic Wait cycles. The wait cycles are used to give the interrupt priority daisy chain (or other priority resolution device) time to settle before the identifier is read. Additional automatic wait states can be generated by programming the Bus Timing and Control register.

The interrupt acknowledge transaction is indicated by an $\overline{M1}$ assertion without $\overline{MREQ}$ during the first cycle. During this transaction the $\overline{IORQ}$ signal becomes active during the third cycle to indicate that the interrupting device can place an 8-bit vector on the bus. It is captured from the AD lines on the falling clock edge just before $\overline{IORQ}$ is raised High.

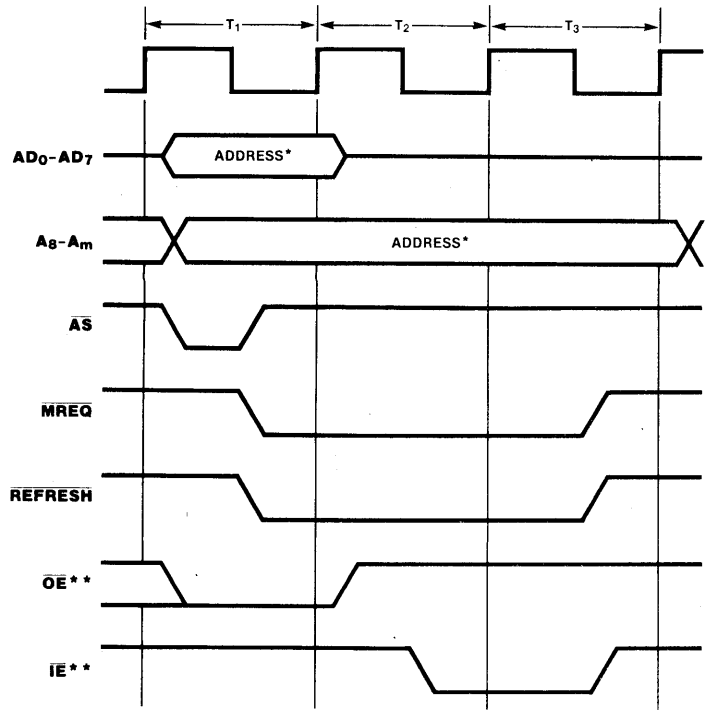
There are two places where the $\overline{WAIT}$ line is sampled and, thus, where a wait cycle can be inserted by external circuitry. The first serves to delay the falling edge of $\overline{IORQ}$ to allow the daisy chain a longer time to settle, and the second serves to delay the point at which the vector is read.

Refresh Transactions. A memory refresh transaction (Figure 45) is generated by the Z800 refresh mechanism and can occur immediately after the final clock cycle of any other transaction. The memory refresh counter's 10-bit address is output on AD_0–AD_9 during the normal time for addresses. The $\overline{RFSH}$ line is activated with $\overline{MREQ}$. This transaction can be used to generate refreshes for dynamic RAMs.



* AD_1 and AD_2 indicate type of interrupt being acknowledged.
** Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 44. Maskable Interrupt Acknowledge Sequence



*10 least-significant bits are Refresh address, the rest are undefined.
** Z8208 only.
m = 18 for Z8108, 23 for Z8208.

Figure 45. Refresh Timing

Requests

There are three kinds of request signals that the Z800 MPU supports. These are:

- Interrupt requests, which another device initiates and the CPU accepts and acknowledges.
- Bus requests, which an external potential bus master initiates and the Z800 MPU accepts and acknowledges.
- Global bus requests, which the CPU or on-chip DMA initiates to acquire a global System bus.

When a request is made, it is answered according to its type: for interrupt requests, an Interrupt Acknowledge transaction is initiated; for bus requests, an Acknowledge signal is sent; for global bus requests, an Acknowledge signal is received.

Interrupt Requests. The Z800 CPU supports two types of interrupt, maskable and nonmaskable ($\overline{NMI}$). The Interrupt Request line of a device that is capable of generating an interrupt can be tied to the $\overline{NMI}$ or maskable interrupt request lines. Several devices can be connected to one pin with the devices arranged in a priority daisy chain. However, all Z80 family peripherals should be on the same line (or no nesting of interrupts among different lines. The CPU uses different protocols for handling requests on the $\overline{NMI}$ pin than the protocol used for maskable interrupt pins. The sequence of events shown below should be followed:

Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge-detected, and the internal $\overline{NMI}$ latch is set. At the beginning of the last clock cycle in the last internal processor clock cycle of any instruction, the interrupt inputs are sampled along with the state of the internal $\overline{NMI}$ latch.

If a maskable interrupt is requested and the Master Status register indicates that requests on that line are to be accepted, the next possible bus transaction is the Interrupt Acknowledge transaction, which results in information from the highest-priority interrupting device being read off the AD lines. This data is used to initiate the interrupt service routine. For a nonmaskable interrupt request, the hexadecimal constant 0066 is used to initiate the interrupt service routine, except in mode 3.

Bus Requests. To generate transactions on the bus, a potential bus master (such as the DMA Controller) must gain control of the bus by making a bus request. A bus request is initiated by pulling $\overline{\text{BUSREQ}}$ Low. Several bus requesters may be wire ORed to the $\overline{\text{BUSREQ}}$ pin; priorities are resolved externally to the CPU, usually by a priority daisy chain.

The asynchronous $\overline{\text{BUSREQ}}$ signal generates an internal $\overline{\text{BUSREQ}}$, which is synchronous. If the external $\overline{\text{BUSREQ}}$ is Low at the beginning of any machine cycle, the internal $\overline{\text{BUSREQ}}$ causes the Bus Acknowledge line ($\overline{\text{BUSACK}}$) to be asserted after the current machine cycle is completed. (Exceptions are the TSET instruction where the read-modify-write cycle is atomic and DMA transfer in burst or continuous mode). The CPU then enters Bus Disconnect state and gives up control of the bus. All CPU Output pins, except $\overline{\text{BUSACK}}$, are 3-stated.

The CPU regains control of the bus after $\overline{\text{BUSREQ}}$ rises. Any device desiring control of the bus must wait at least two bus cycles after $\overline{\text{BUSREQ}}$ has risen before pulling it down again.

The on-chip DMA channels have higher priority than external devices requesting the bus via $\overline{\text{BUSREQ}}$.

EXTERNAL INTERFACE (Z-BUS)

Features

- 16-bit data bus
- Multiplexed address/data lines
- Supports high-speed burst mode transfers
- Provides EPA interface

Pin Descriptions

A. *Address* (output, active High, 3-state). These address lines carry I/O addresses and memory addresses during bus transactions. Of the eight lines, only three are available on the 40-pin version.

AD. *Address/Data* (bidirectional, active High, 3-State). These 16 multiplexed address and data lines carry I/O addresses, memory addresses, and data during bus transactions.

$\overline{\text{AS}}$. *Address Strobe* (output, active Low, 3-state). The rising edge of Address Strobe indicates the beginning of a transaction and shows that the address, status, R/$\overline{\text{W}}$ and B/$\overline{\text{W}}$ signals are valid.

$\overline{\text{BUSACK}}$. *Bus Acknowledge* (output, active Low). A Low on this line indicates that the CPU has relinquished control of the bus in response to a bus request.

$\overline{\text{BUSREQ}}$. *Bus Request* (input, active Low). A Low on this line indicates that an external bus requester has obtained or is trying to obtain control of the bus.

B/$\overline{\text{W}}$. *Byte/Word* (output, Low = Word, 3-state). This signal indicates whether a byte or a word of data is to be transmitted during a transaction.

CLK. *Clock Output* (output). The frequency of the processor timing clock is derived from the oscillator input (external oscillator) or crystal frequency (internal oscillator) by dividing the crystal or external oscillator input by two. This clock is further divided by one, two or four (as programmed), and then output on this line.

$\overline{\text{DS}}$. *Data Strobe* (output, active Low, 3-state). This signal provides timing for data movement to or from the bus master.

$\overline{\text{INT}}$. *Maskable Interrupts* (input, active Low). A Low on these lines requests an interrupt. Of the three lines, only one is available on the 40-pin version.

$\overline{\text{NMI}}$. *Nonmaskable Interrupt* (input, falling-edge activated). A High-to Low transition on this line requests a Nonmaskable Interrupt.

$\overline{\text{RESET}}$. *Reset* (input, active Low). A Low on this line resets the CPU.

R/$\overline{\text{W}}$. *Read/Write* (output, Low = Write, 3-state). This signal determines the direction of data transfer for memory, I/O, or EPU transfer transactions.

ST. *Status* (output, active High, 3-state). These four lines indicate the type of transaction occuring on the bus and give additional information about the transaction.

$\overline{\text{WAIT}}$. *Wait* (input, active Low). A Low on this line indicates that the responding device needs more time to complete a transaction.

XTALI. *Clock/Crystal Input* (time-base input). Connects a series-resonant crystal or an external single-phase clock to the on-chip clock oscillator.

XTALO. *Crystal Output* (time-base output). Connects a series-resonant crystal to the on-chip clock oscillator.

+5 V. *Power Supply Voltage.* (+5 nominal).

GND. *Ground.* Ground reference.

The following lines are available on the 64-pin device version only:

$\overline{\text{DMASTB}}$. *DMA Flyby Strobe* (output, active Low). These lines select peripheral devices for DMA flyby transfers.

CTIN. *Counter/Timer Input* (input, active High). These lines receive signals from external devices for the counter-timers.

CTIO. *Counter/Timer I/O* (bidirectional, active High, 3-state). These I/O lines transfer signals between the counter/timers and external devices.

$\overline{IE}$. *Input Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is toward the CPU.

$\overline{OE}$. *Output Enable* (output, active Low, 3-state). A Low on this line indicates that the direction of transfer on the Address/Data lines is away from the MPU.

$\overline{PAUSE}$. *CPU Pause* (input, active Low, Z8216 only). While this line is Low the CPU refrains from transferring data to or from an Extended Processing Unit in the system or from beginning the execution of an instruction.

RX. *UART Receive* (input, active High). This line receives serial data at standard TTL levels.

$\overline{RDY}$. *DMA Ready* (input, active Low). These lines are monitored by the DMA channels to determine when a peripheral device associated with a DMA channel is ready for a read or write operation. When a DMA channel is enabled to operate, its Ready line indirectly controls DMA activity; the manner in which DMA activity is controlled by the line varies with the operating mode (single-transaction, burst or continuous).

TX. *UART Transmit* (output, active High). This line transmits serial data at standard TTL levels.

Bus Operations

Two kinds of operations can occur on the system bus: transactions and request. At any given time, one device (either the CPU or a bus requester) has control of the bus and is known as the bus master. A transaction is initiated by the bus master and is responded to by some other device on the bus. Only one transaction can proceed at a time; nine kinds of transactions can occur:

Burst Memory. These transactions are used to transfer four words of instructions from the memory to the CPU.

DMA Flyby. This transaction is used by the DMA peripheral to transfer data between an external peripheral and memory.

EPU Transfer. This transaction is used to transfer data between the CPU and an EPU.

Halt. This transaction is used to indicate that the CPU is executing the Halt instruction.

Internal Operation. These transactions do not transfer data.

Interrupt Acknowledge. This transaction is used by the CPU to acknowledge an external interrupt request and to transfer additional information from the interrupting device.

I/O. This transaction is used by the bus master to transfer data to or from an external peripheral.

Memory. This transaction is used by the bus master to transfer data to or from a memory location.

Refresh. These transactions by the refresh mechanism do not transfer data; they refresh dynamic memory.

Only the bus master can initiate transactions. A request, however, can be initiated by a device that does not have control of the bus. Two types of requests can occur:

Bus. This request is used to request control of the bus to initiate transactions.

Interrupt. This request is used to request the servicing by the CPU.

When an interrupt or bus request is made, it is answered according to its type: for an externally generated interrupt request, an Interrupt Acknowledge transaction is initiated by the CPU; for bus requests, the MPU enters Bus Disconnect state, relinquishes the bus, and activates an acknowledge signal.

Transactions

Data transfers to and from the Z800 MPU are accomplished through the use of transactions.

All transactions start with Address Strobe ($\overline{AS}$) being driven Low and then raised High by the Z800 MPU. On the rising edge of $\overline{AS}$, the Status lines ST_0–ST_3 are valid; these lines indicate the type of transaction being initiated (Table 8); seven types of transactions are discussed in the sections that follow. Associated with the status lines are two other lines that become valid at this time: R/$\overline{W}$, and B/$\overline{W}$.

Table 8. Status Table

Code	Meaning
0000	Internal operation
0001	Refresh
0010	I/O transaction
0011	Halt
0100	Interrupt acknowledge line A
0101	Interrupt acknowledge (nonmaskable)
0110	Interrupt acknowledge line B
0111	Interrupt acknowledge line C
1000	Memory Reference (cachable)
1001	Memory Reference (non-cachable)
1010	Memory–EPU transfer
1011	Reserved
1100	EPU Instruction fetch
1101	EPU Instruction fetch (first word)
1110	EPU–CPU transfer
1111	Test and Set (data transfers)

If the transaction requires an address, it is valid on the rising edge of $\overline{AS}$. No address is required for EPU-CPU transfer and Internal Operation transactions; the contents of the A and AD lines while $\overline{AS}$ is asserted are undefined. If an address is generated, the $\overline{OE}$ signal is also activated.

The Z-BUS MPUs use Data Strobe ($\overline{DS}$) to time the actual data transfer. (Note that Refresh, Halt, and Internal Operation transactions do not transfer any data and thus do not activate $\overline{DS}$.) For write operations (R/$\overline{W}$ = Low), a Low on $\overline{DS}$ indicates that valid data from the bus master is on the AD lines. The Output Enable continues to be asserted until $\overline{DS}$ is deasserted. For read operations (R/$\overline{W}$ = High), the bus master makes AD Lines 3-state, deasserts $\overline{OE}$ and asserts $\overline{IE}$ after driving $\overline{DS}$ Low so that the addressed device can put its data on the bus. The bus master samples this data on the falling clock edge just before raising $\overline{DS}$ and $\overline{IE}$ High.

Wait. The Wait line is sampled on the falling clock edge when data is sampled by the Z800 MPU (Read), or the falling clock edge before $\overline{DS}$ rises (Read or Write). If $\overline{WAIT}$ is Low, another cycle is added to the transaction before data is sampled or $\overline{DS}$ rises. In this added cycle, and all subsequent cycles added when $\overline{WAIT}$ is Low, $\overline{WAIT}$ is again sampled on the falling clock edge and, if it is Low, another cycle is added to the transaction. In this way, the transaction can be extended to an arbitrary length by external circuitry to accommodate (for example) slow memories or I/O devices that are not yet ready for data transfer. Automatic insertions of wait states by the CPU or on-chip DMA channels can be programmed by setting fields in the Bus Timing and Control register and Bus Timing and Initialization register to indicate the number to be inserted.

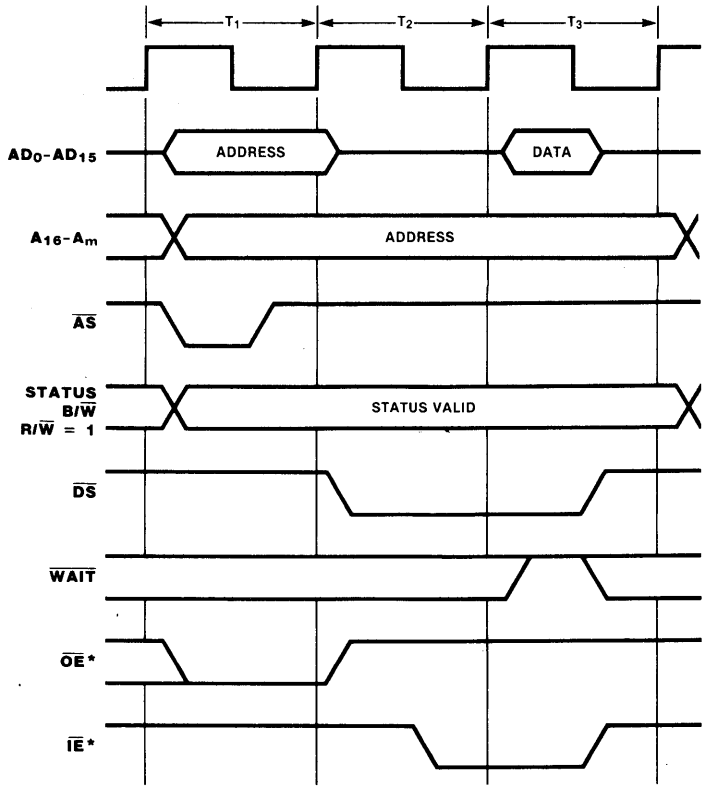
Memory Transactions. Memory transactions move data to or from memory when a bus master makes a memory access. Thus, they are generated during program execution to fetch instructions from memory and to fetch and store memory data. They are also generated to store old program status and fetch new program status during interrupt and trap handling and after reset.

A memory transaction is three bus cycles long unless extended when $\overline{WAIT}$ is asserted, as explained above in the Wait section. The status pins, besides indicating a memory transaction, give the following information:

- Whether the memory access is cacheable (1000) or noncacheable (1001) information.
- Whether the data for the access is supplied (written) or captured (read) by an Extended Processing Unit (1010).
- Whether the information being fetched from memory is an EPA template (1100, 1101).
- Whether the memory access is part of the read-modify-write operation (Test and Set) (1111).

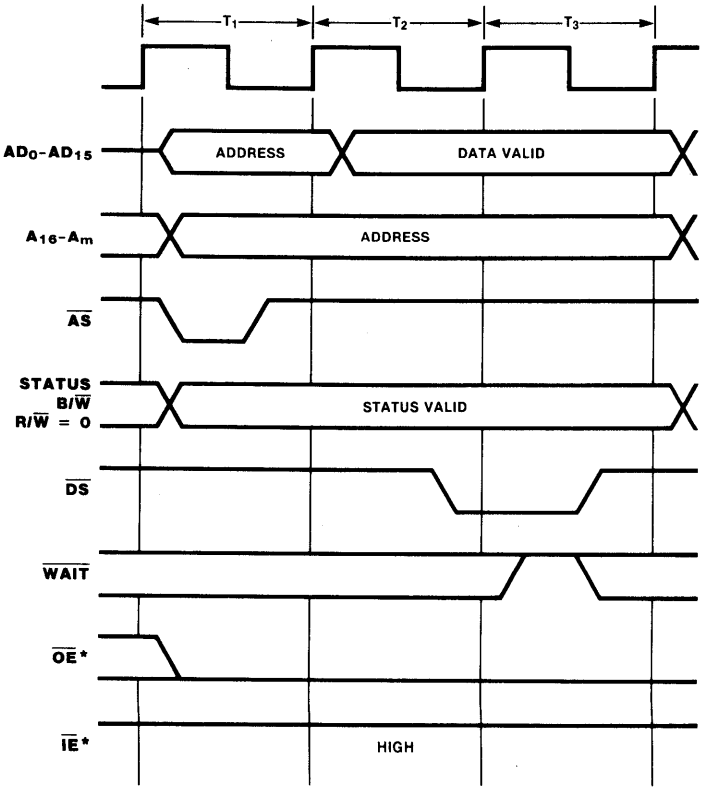
Bytes transferred to or from odd memory (address bit 0 is 1) locations are always transmitted on lines AD_0–AD_7 (bit 0 on AD_0). Bytes transferred to or from even memory locations (address bit 0 = 0) are always transmitted on lines AD_8–AD_{15} (bit 0 on AD_8). For byte reads (B/$\overline{W}$ High, R/$\overline{W}$ High), the CPU or on-chip DMA channel uses only the byte whose address it put out on the bus. For byte writes (B/$\overline{W}$ High, R/$\overline{W}$ Low), the memory should store only the byte whose address was output. During byte memory writes, the CPU (or on-chip DMA channel in non-Flyby transactions) places the same byte on both halves of the bus, and the proper byte must be selected by testing A_0. For word transfers, (R/$\overline{W}$ = Low), all 16 bits are captured by the CPU or DMA channel (Read: R/$\overline{W}$ = High) or stored by the memory (Write: R/$\overline{W}$ = Low). For these transactions (either memory or I/O) the bytes of data appear swapped on the bus with the most significant byte on AD_7–AD_0 and the least significant byte on AD_{15}–AD_8.

Memory transaction timings are shown in Figures 46-50.



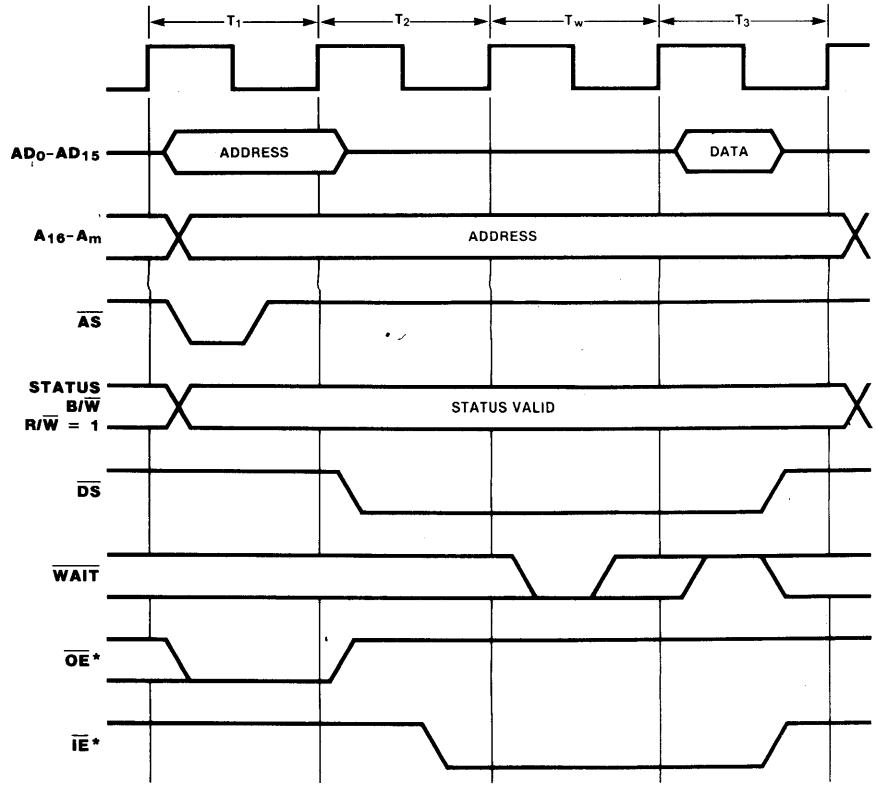
*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 46. Memory Read Timing



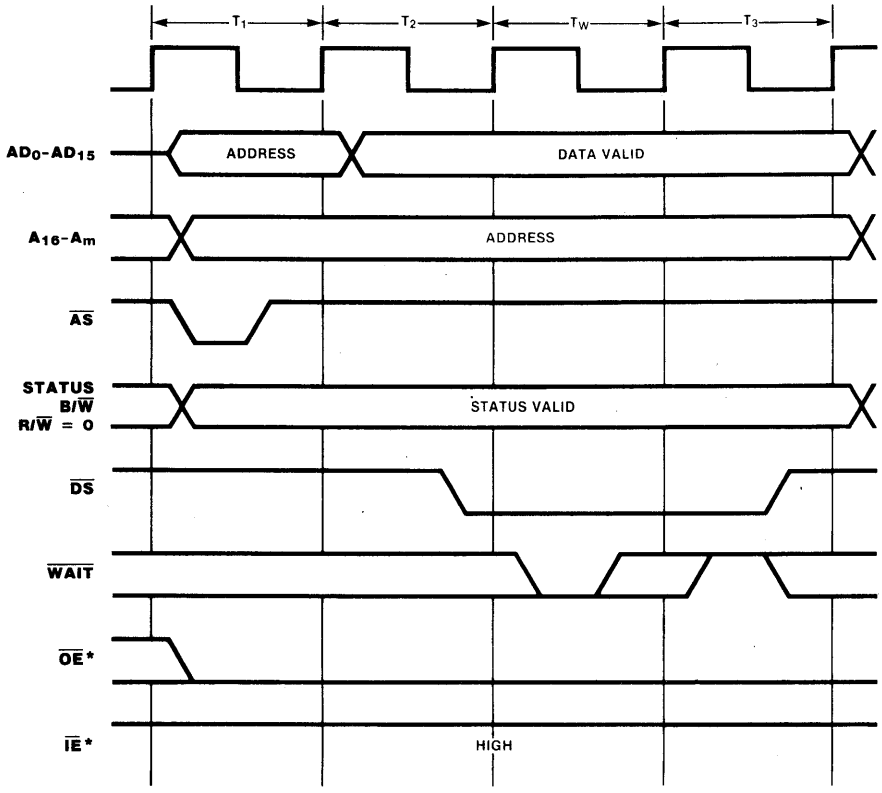
*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 47. Memory Write Timing



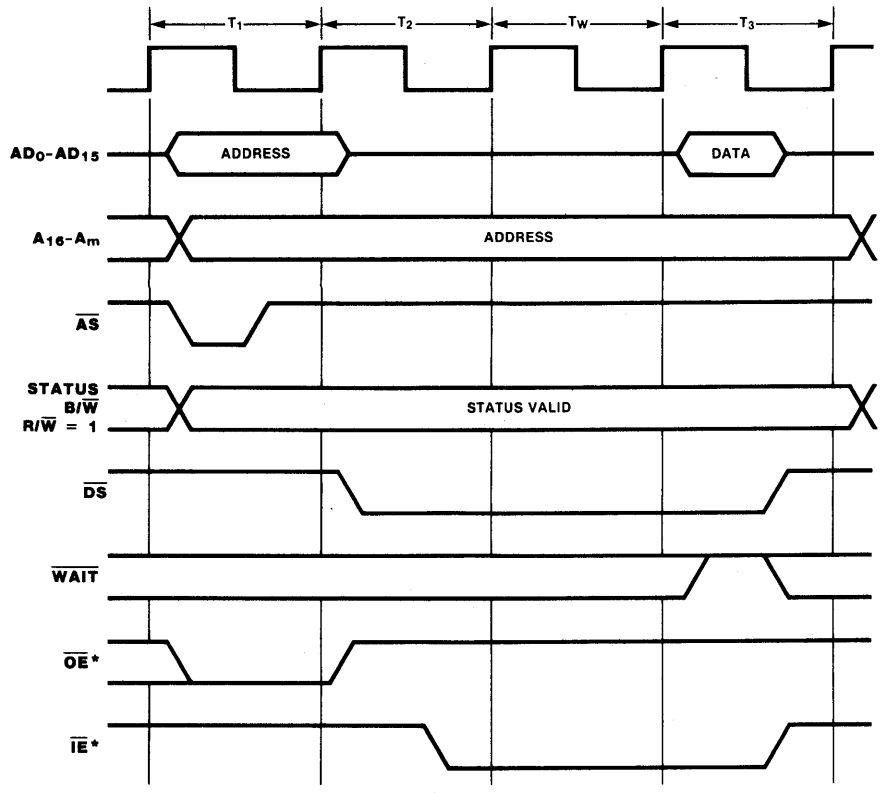
*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 48. Memory Read Timing with External Wait Cycle



*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 49. Memory Write Timing with External Wait Cycle



*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 50. Memory Read Timing with Internal Wait Cycle

Burst Memory Transactions. Burst memory transactions use multiple Data Strobes associated with a single Address Strobe. The CPU uses burst transactions to read four consecutive words in four data transactions. The address of the first word read during a burst transaction has zeros in the three least significant bits. Control bits in the Cache Control register indicate whether or not portions of the memory system can support burst transactions.

The CPU uses burst mode reads only for fetching instructions. If an instruction is to be fetched from a location within a half of physical memory that supports burst transactions, the CPU reads the eight bytes that contain the first byte of the instruction. (EPA template fetches and the RETI instruction do not use the burst transaction.)

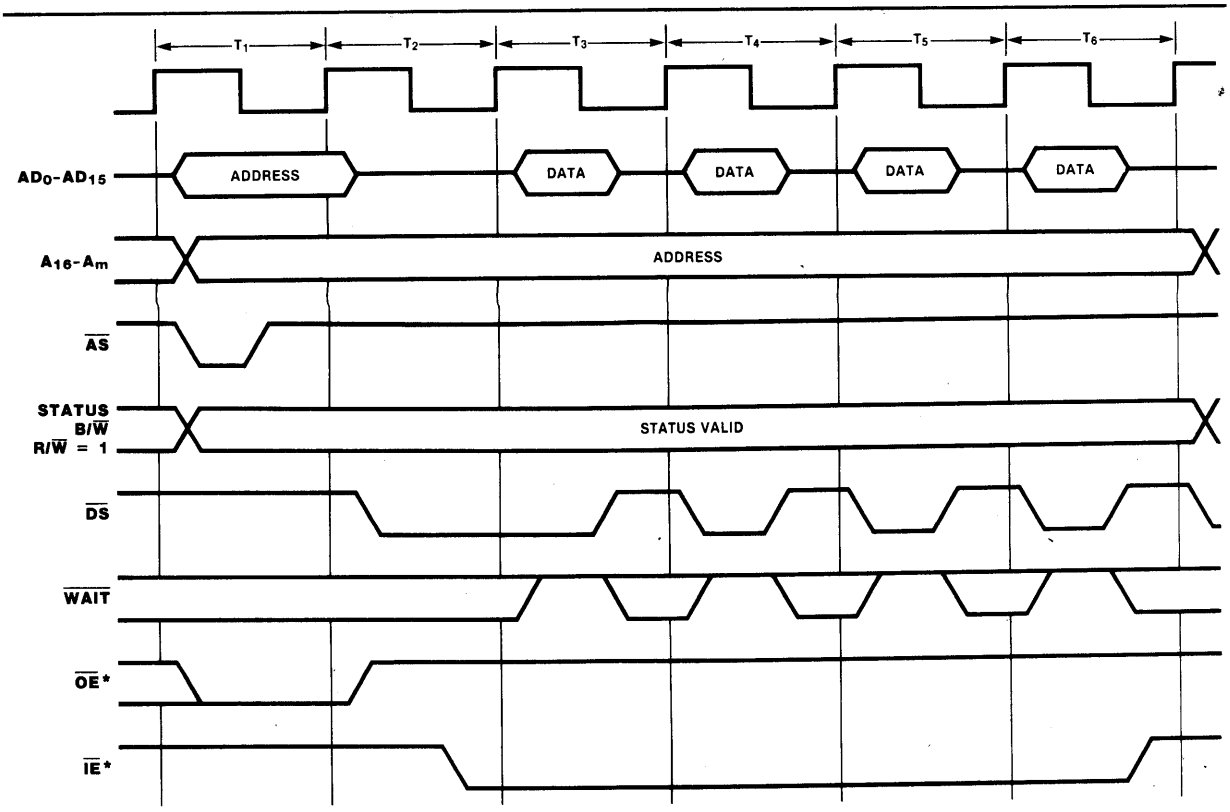
Timing for the first data transfer during a burst transaction is identical to that for a single memory read, including the automatic insertion of wait states, except there are four T_3 states. Subsequent data transfers do not include automatic wait states. On the first data transfer, if $\overline{WAIT}$ is sampled active then it is sampled again every bus clock cycle until it is inactive, at which time the data is read from the bus. Burst memory read timing is shown in Figure 51.

Internal Operation and Halt Transactions. Two types of bus transactions made by the CPU do not transfer data: Internal Operations and Halt transactions. These transactions look like a memory transaction, except that $\overline{DS}$ remains High and no data is transferred.

For the Internal Operation transaction (Figure 52), the Address lines contain arbitrary data when $\overline{AS}$ goes High. The R/$\overline{W}$ line indicates Read (High) and the Status lines indicate Internal Operation (0000).

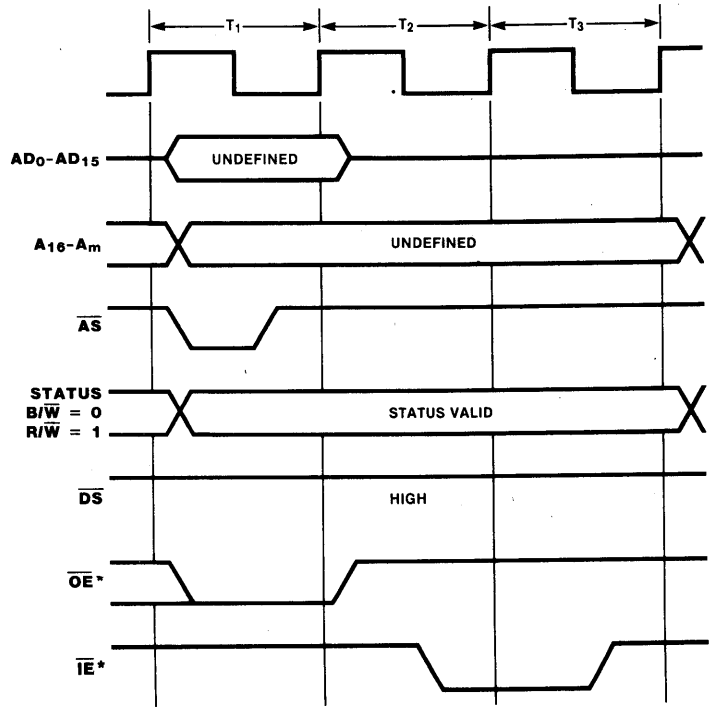
A Halt transaction (Figure 53) is generated when the CPU executes a Halt instruction or when a fatal sequence of traps and bus errors occurs. The address placed on the AD lines is the location of the Halt instruction or the instruction that initiated the fatal sequence of traps and errors. The Status lines indicate a Halt transaction (0011).

$\overline{WAIT}$ is not sampled during the Internal Operation or Halt transaction.



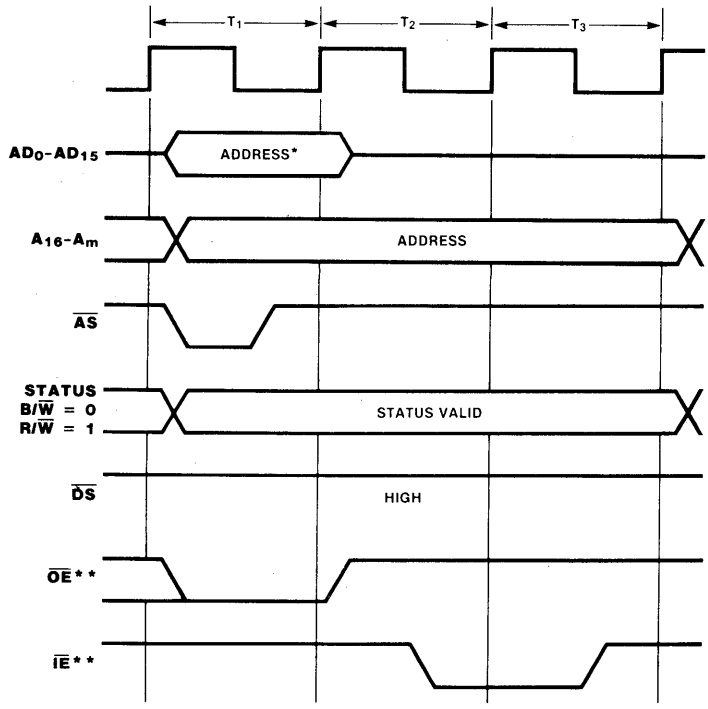
*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 51. Burst Memory Read Timing



*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 52. Internal Operation Timing



*Address of Halt Instruction.
**Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 53. Halt Timing

I/O Transactions. I/O transactions (Figures 54 and 55) move data to or from peripherals and are generated during the execution of I/O instructions. I/O transactions to on-chip peripheral devices do not generate external bus transactions.

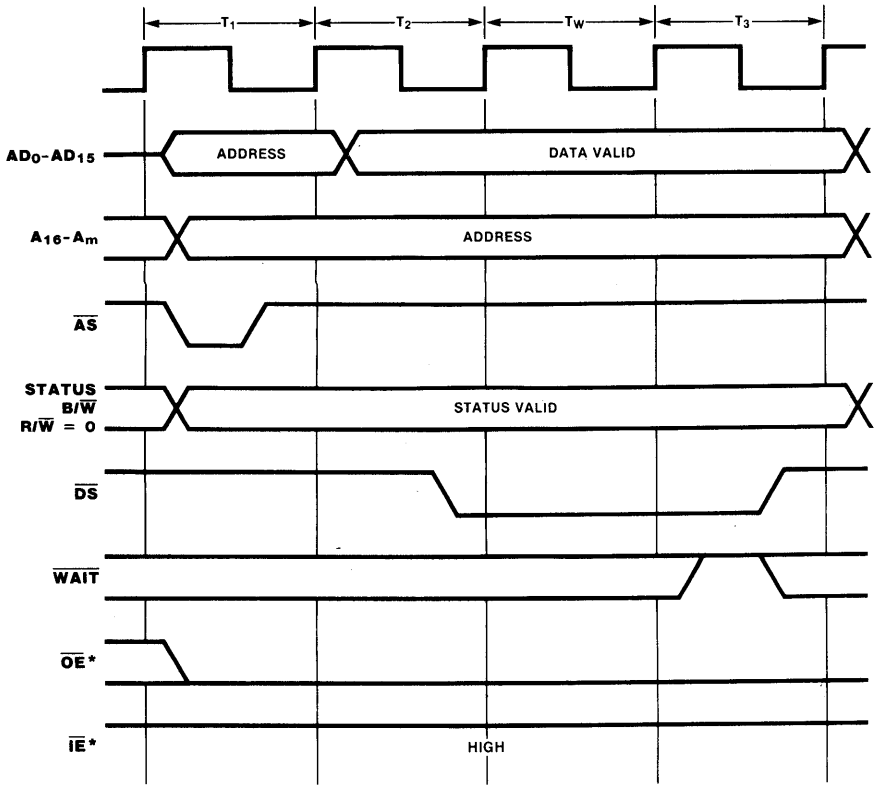
I/O transactions are four bus cycles long at a minimum, and they can be lengthened by the addition of wait cycles either automatically generated as indicated in the Bus Timing and Control register or generated by an external device. The extra clock cycles allow for slower peripheral operation.

The status lines indicate that the access is an I/O transaction (0010). The I/O address is found on AD_0–AD_{15} and A_{16}–A_{23}.

Byte data (B/$\overline{W}$ = High) is transmitted on AD_0–AD_7. This allows peripheral devices to attach to only eight of the AD lines. The Read/Write line (R/$\overline{W}$) indicates the direction of the data transfer: peripheral-to-CPU (Read: R/$\overline{W}$ = High) or CPU-to-peripheral (Write: R/$\overline{W}$ = Low).

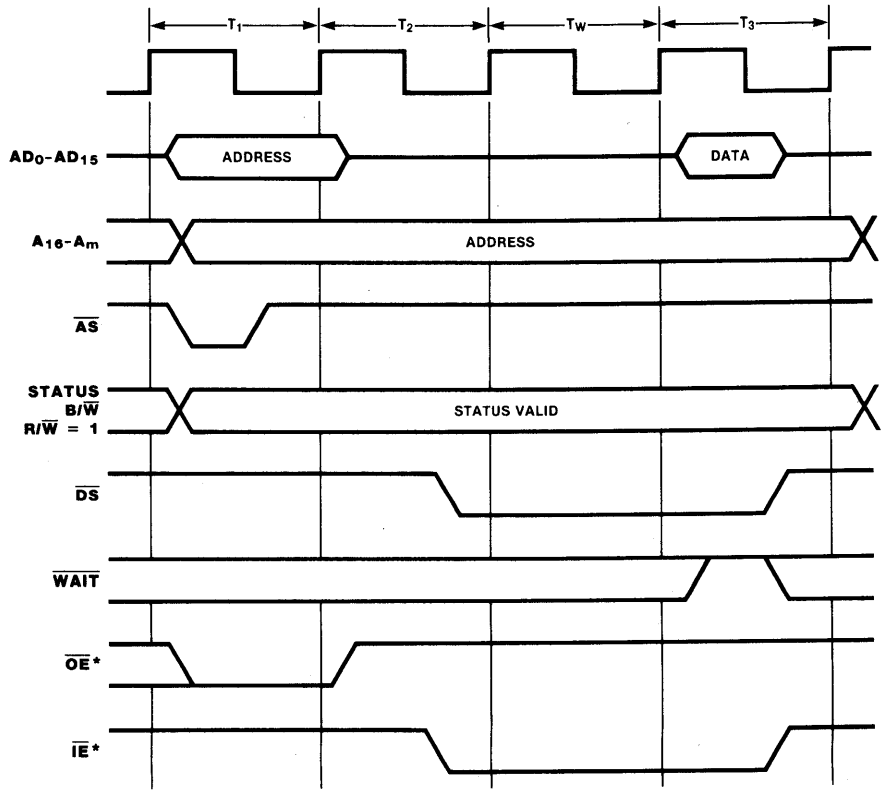
Interrupt Acknowledge Transactions. These transactions (Figure 56) acknowledge an interrupt and read an identifier from the device that generated the interrupt. Interrupt Acknowledge transactions are generated automatically by the hardware when an interrupt is detected.

These transactions are five cycles long at a minimum, with at least two automatic wait cycles, although others can be added by programming the Bus Timing and Control register. The wait cycles are used to give the interrupt priority daisy chain (or other priority resolution device) time to settle before the identifier is read.



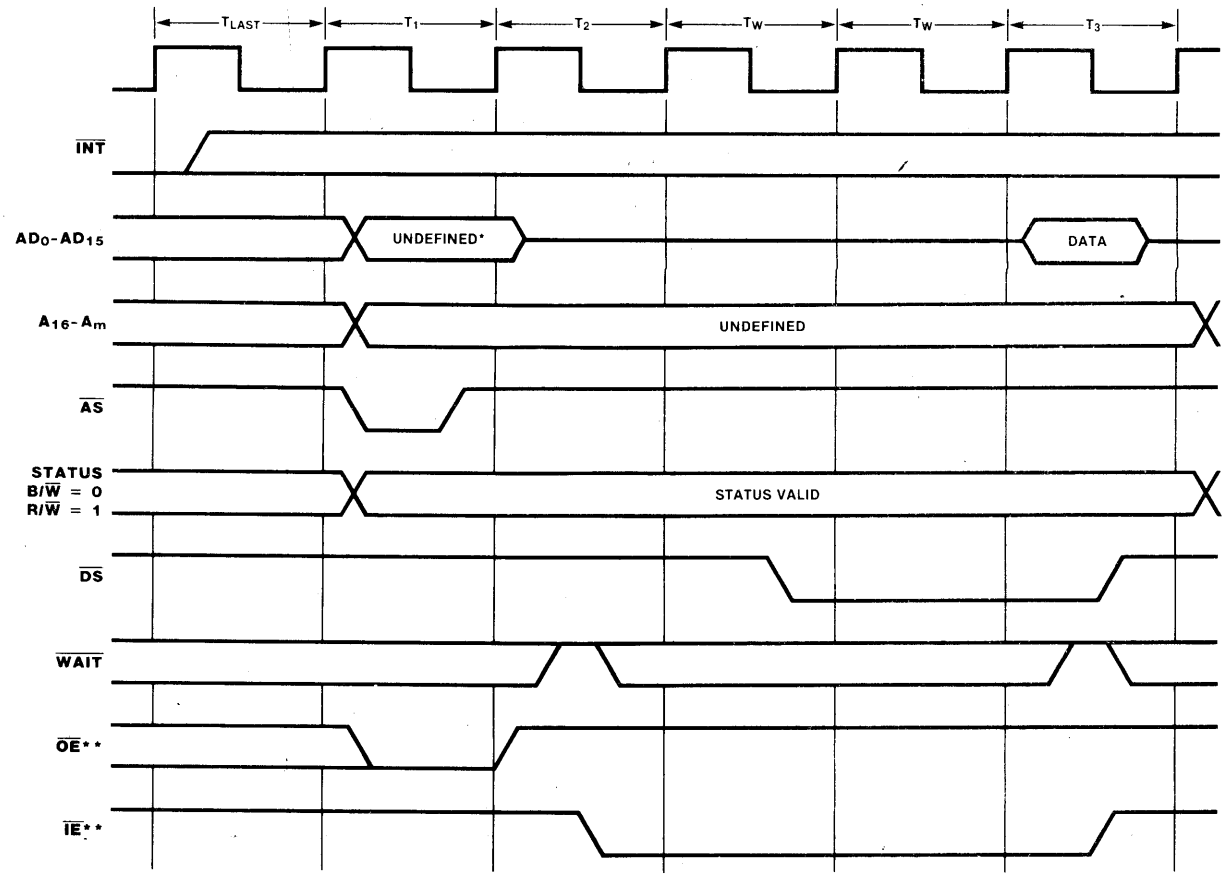
*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 54. I/O Write Timing



*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 55. I/O Read Timing



* AD_1 and AD_2 indicate the type of interrupt being acknowledged.
** Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 56. Interrupt Acknowledge Timing

The Status lines identify the type of interrupt that is being acknowledged. The possibilities are nonmaskable interrupt (0101) and the three external interrupt acknowledges (0100, 0101 and 0111). No address is generated; the contents of the bus are undefined when $\overline{AS}$ rises. The R/$\overline{W}$ line indicates Read (High), and the B/$\overline{W}$ line indicates Word (Low).

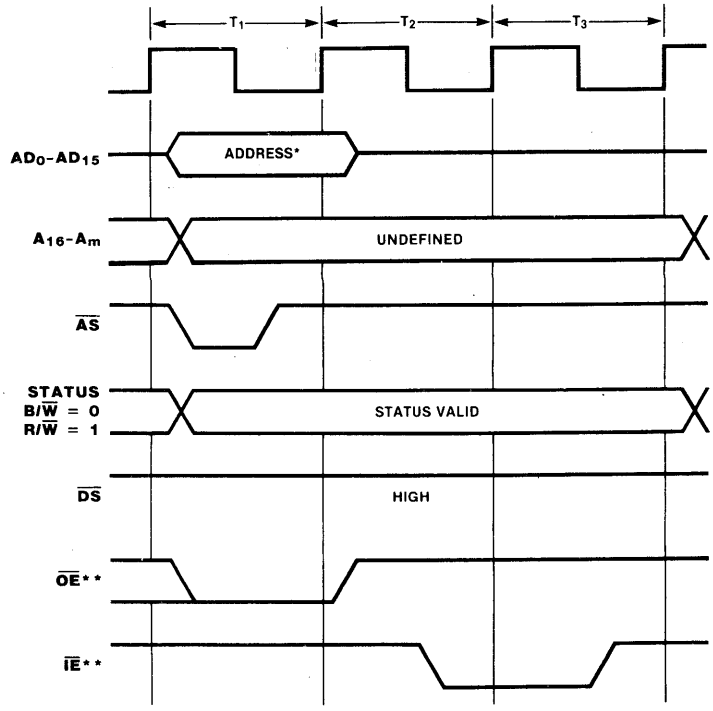
The only item of data transferred is the identifier that is captured from the AD lines on the falling clock edge just before $\overline{DS}$ is raised High. The length of time that $\overline{DS}$ is asserted is identical with I/O timing programmed in the Bus Timing and Control register.

There are two places where $\overline{WAIT}$ is sampled and thus a wait cycle can be inserted by external devices. The first place serves to delay the falling edge of $\overline{DS}$ to allow the daisy chain a longer time to settle, and the second place serves to delay the point at which data is read.

Refresh Transactions. A memory Refresh transaction (Figure 57) is generated by the refresh mechanism and can come immediately after the final clock cycle of any other transaction. The memory refresh counter's 10-bit address is output on the low order 10 bits of the bus during the first cycle of the transaction. The contents of the rest of the bus are undefined. The Status lines indicate Refresh (0001). This transaction can be used to generate refreshes for dynamic RAMs. Refreshes may occur while the CPU is in the Halt or Fatal state.

CPU-Extended Processing Unit Interaction

The Z800 CPU with a Z-BUS interface and $\overline{PAUSE}$ input line (i.e., the Z8216) and one or more Extended Processing Units (EPUs) work together like a single CPU component, with the CPU providing address, status, and timing signals and the EPU supplying and capturing data. The EPU monitors the status and timing signals output by the CPU so that it knows when to participate in a memory transaction; for EPU to memory transfers, the CPU puts its AD lines in 3-state while $\overline{DS}$ is Low, so that the EPU can use them.



*10 least-significant bits are Refresh address.
**Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 57. Memory Refresh Timing

In order to know which transaction it is to participate in, the EPU must track the following sequence of events:

- When the CPU fetches the first word of an EPA instruction template from memory (ST_3–ST_0 = 1101), the EPU must also capture the instruction returned by the memory. The template has an ID field that indicates whether or not the EPU is to execute the instruction. Because there is no alignment restriction on EPA templates, the ID field can not be in the first word fetched.
- The next non-refresh transaction by the CPU is the fetching the second word of the instruction (ST_3–ST_0 = 1100). The EPU must also capture this word. If the template is not aligned, a third fetch is made (ST_3–ST_0 = 1100).
- If the instruction involves a read or write to memory, then transfers of data between memory and the EPU (ST_3–ST_0 = 1010) are the next non-refresh transactions performed by the CPU. The EPU must supply the data (Write: R/$\overline{W}$ = Low) or capture the data (Read: R/$\overline{W}$ = High) for each transaction, just as if it were part of the CPU. In both cases, the CPU 3-states its AD lines while data is being transferred ($\overline{DS}$ Low).
- If the instruction involves a transfer from the EPU to the Z800 MPU, the next non-refresh transaction is the CPU transferring data between the EPU and CPU (ST_3–ST_0 = 1110).

In order to follow this sequence, an EPU has to monitor the status lines to verify that the transaction it is monitoring on the bus was generated by the CPU. In a multiple EPU system, there is no indication on the bus as to which EPU is cooperating with the CPU at any given time. This must be determined by the EPUs from the EPA templates they capture.

When an EPU begins to execute an extended instruction, the CPU can continue fetching and executing instructions. If the EPU wishes to halt the CPU from executing another instruction or bus transaction, the EPU must activate the $\overline{PAUSE}$ line to stop the CPU until the EPU is ready for subsequent CPU activity. This mechanism is used to synchronize CPU-EPU activity.

EPU Transfer Transactions. These transactions (Figures 58-60) move data between the CPU and an EPU, thus allowing the CPU to transfer data to or from an EPU or to read or write an EPU's status registers. They are generated during the execution of the EPA instruction.

EPU-to-Memory transfers are five cycles unless extended by Wait. Memory to EPU transfers are three cycles unless extended by $\overline{WAIT}$.

EPU-CPU transfer transactions have the same form as I/O transactions and thus are four clock cycles long, unless extended by $\overline{WAIT}$. Although $\overline{AS}$ is asserted, no address is generated and the contents of the bus are undefined; only one status code is used (1110).

In a multiple EPU system, the EPU that is to participate in a transaction is selected implicitly by the ID code in the EPU template, rather than by an address. The Read/Write line (R/$\overline{W}$ = High) indicates the direction of the data transfer into the CPU.

Requests

The Z800 MPU supports three types of request signal. These are:

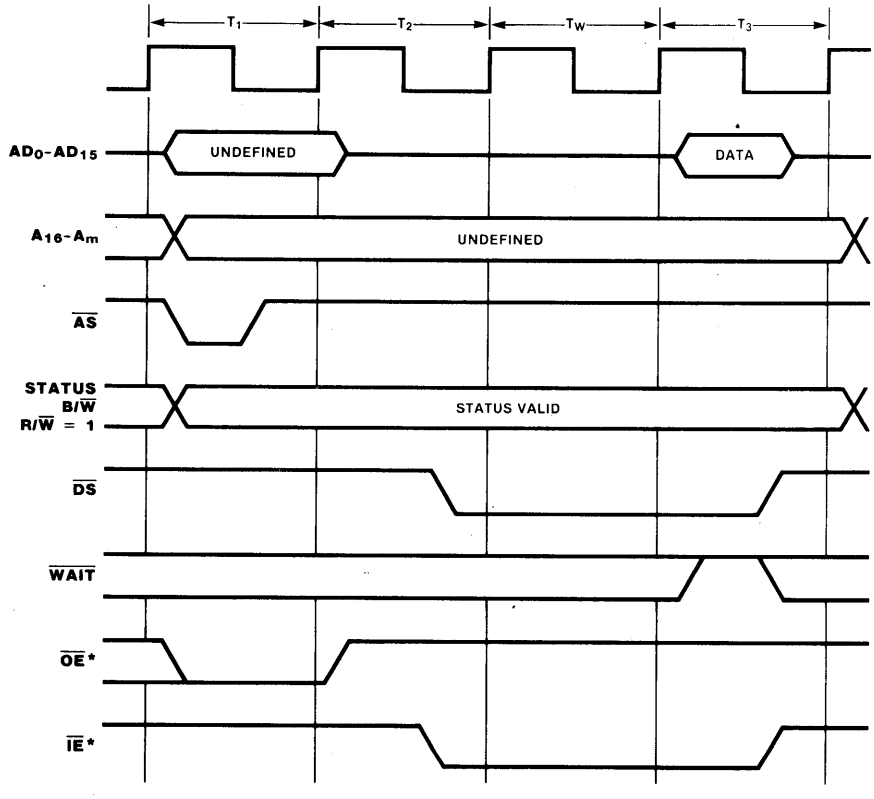
- Interrupt requests, which another device initiates and the CPU accepts and acknowledges.
- Bus requests, which an external potential bus master initiates and the CPU accepts and acknowledges.
- Global bus requests, which the CPU or on-chip DMA initiates to acquire a global system bus.

When a request is made, it is answered according to its type: for interrupt requests, an Interrupt Acknowledge transaction is initiated by the CPU; for bus requests, an acknowledge signal is sent; for global bus request, an acknowledge signal is received.

Interrupt Requests. The Z800 MPU supports two types of external interrupt, maskable and nonmaskable ($\overline{NMI}$). The Interrupt Request line of a device that is capable of generating an interrupt may be tied to any of the interrupt pins. Several devices can be connected to one pin, with the devices arranged in a priority daisy chain. The CPU uses the same protocol for handling requests on these pins. The sequence of events is given below:

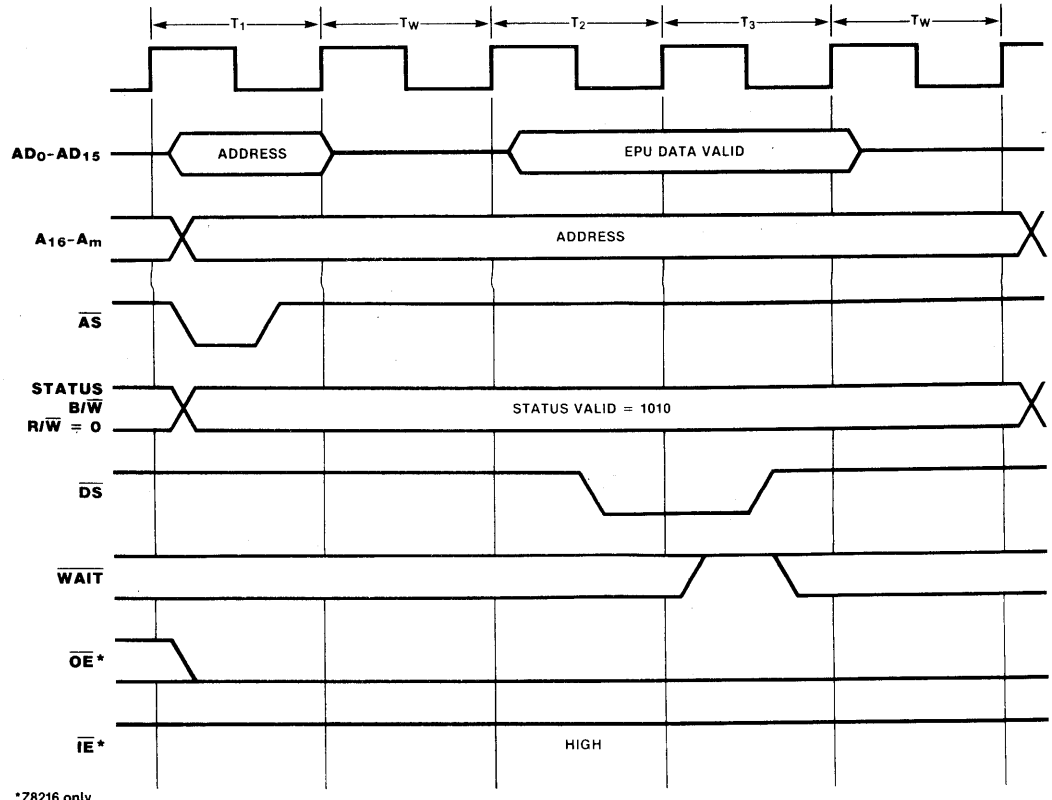
Any High-to-Low transition on the $\overline{NMI}$ input is asynchronously edge-detected, and the internal $\overline{NMI}$ latch is set. At the beginning of the last processor clock cycle of any instruction, the interrupt inputs are sampled along with the state of the internal $\overline{NMI}$ latch.

If a maskable interrupt is requested and the Master Status register indicates that requests on that line are to be accepted, or if the $\overline{NMI}$ latch is set, the next possible bus transaction is an interrupt acknowledge transaction that results in an identifier from the highest-priority interrupting device being read off the AD lines. This data is



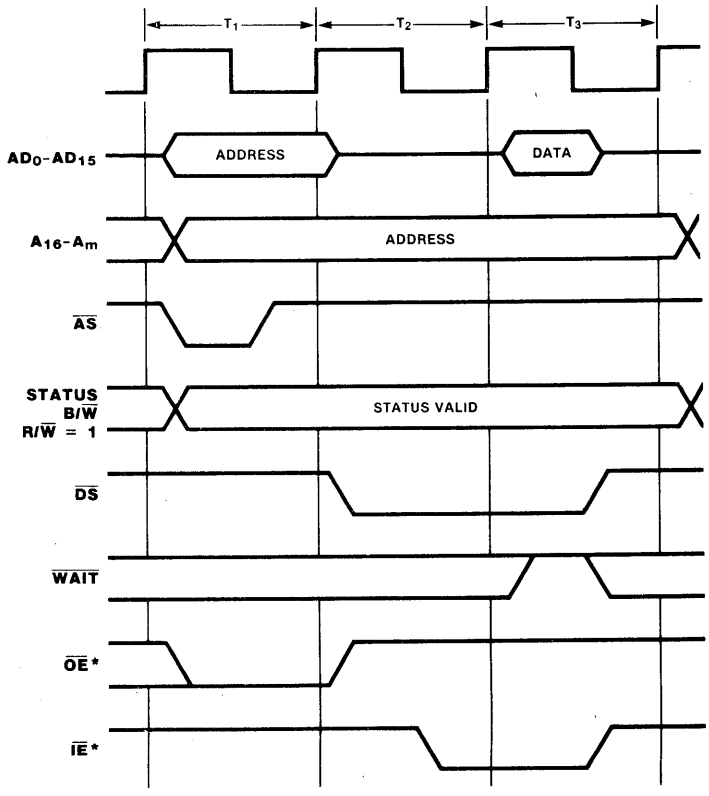
*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 58. EPU to CPU Timing



*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 59. EPU Write to Memory



*Z8216 only.
m = 18 for Z8116, 23 for Z8216.

Figure 60. Memory to EPU Timing

used to initiate the interrupt service routine. For a non-maskable interrupt request in interrupt mode 0, 1, or 2, an interrupt acknowledge transaction is not generated; the hexidecimal constant 0066 is used to initiate the interrupt service routine.

Bus Requests. To generate transactions on the bus, a potential external bus master (such as a DMA Controller) must gain control of the bus by making a bus request. A bus request is initiated by pulling $\overline{\text{BUSREQ}}$ Low. Several bus requesters can be wire ORed to the $\overline{\text{BUSREQ}}$ pin; priorities are resolved externally to the CPU, usually by a priority daisy chain.

The asynchronous $\overline{\text{BUSREQ}}$ signal generates an internal $\overline{\text{BUSREQ}}$, which is synchronous. If the external $\overline{\text{BUSREQ}}$ is Low at the beginning of any processor clock cycle, the internal $\overline{\text{BUSREQ}}$ will cause the bus acknowledge line ($\overline{\text{BUSACK}}$) to be asserted after the current bus transaction is completed or after the write transaction of a TSET instruction. The CPU then enters Bus Disconnect state and gives up control of the bus. All Z800 Output pins except $\overline{\text{BUSACK}}$ are 3-stated.

The on-chip DMA channels have higher priority than the off-chip devices requesting the external bus via $\overline{\text{BUSREQ}}$.

RESET

A hardware reset puts the Z800 MPU into a known state and optionally initializes the Bus Timing and Initialization control register of the Z800 MPU to a system specifiable value. A reset begins at the end of any processor clock cycle if the $\overline{\text{RESET}}$ line is Low. However, if a bus transaction is in progress it is allowed to be completed. A system reset overrides all other operations of the chip, including interrupts, traps and bus requests. A reset should be used to initialize a system as part of the power-up sequence.

Within 128 processor clock cycles of the $\overline{\text{RESET}}$ line becoming Low, the Z800 lines assume their reset values. For either bus, the AD lines are 3-stated, and all control outputs are forced High. While $\overline{\text{RESET}}$ is asserted, the clock output is the processor clock frequency scaled by four. $\overline{\text{RESET}}$ must be held low at least 128 processor clock cycles.

The Reset line is sampled on the rising edge of the clock output during reset. When the Reset line is sampled High (de-asserted), the state of the Wait line is also noted: if $\overline{\text{WAIT}}$ is asserted, then the contents of the AD lines on the falling edge of the clock are used to program the content of the Bus Timing and Initialization register, otherwise the constant 00 hexadecimal is used. If the hardware programming option is used, AD_6 is used to enable the bootstrap via UART option.

After reset, the following control registers are initialized as follows:

- Program Counter, System Stack Pointer, I, and R registers initialized to 0
- Master Status register—initialized to 0, e.g., system mode of operation; single-step mode, Breakpoint-on-Halt and all maskable interrupts disabled
- I/O Page register—I/O page 0 in use
- Stack Limit register cleared
- Refresh register—initialized to 88, e.g., refresh enabled, rate = 32 clock cycles
- Cache Control register—initialized to 00, e.g. cache enabled for program only (associative rather than fixed location); also, all lines invalid
- Memory Management Unit Master Control register—initialized to 0, e.g., translation disabled
- Trap Control register—initialized to 0, e.g., Stack Warning disabled, EPA disabled, I/O not privileged
- All peripheral control registers—peripheral disabled (but see UART bootstrap option)
- Interrupt Status register—Interrupt Mode 0

The following registers are unaffected:

- CPU register file, including user Stack Pointer
- Page Descriptor registers
- Interrupt/Trap Vector Table Pointer register

On the rising edge of $\overline{\text{RESET}}$, if Bus Request is asserted the Z800 MPU will grant the bus before fetching the first instruction from location 0.

After $\overline{\text{RESET}}$ has returned to High, the CPU begins to operate unless the Bootstrap UART feature is utilized.

PIN ASSIGNMENTS

The pin assignments of four versions of the Z800 MPU, the Z8108, Z8208, Z8116 and Z8216 are shown in Figures 61-64 respectively.

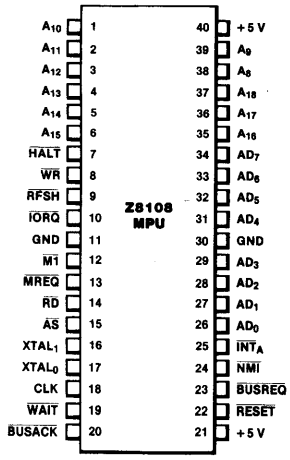


Figure 61. Z8108 Pin Assignments

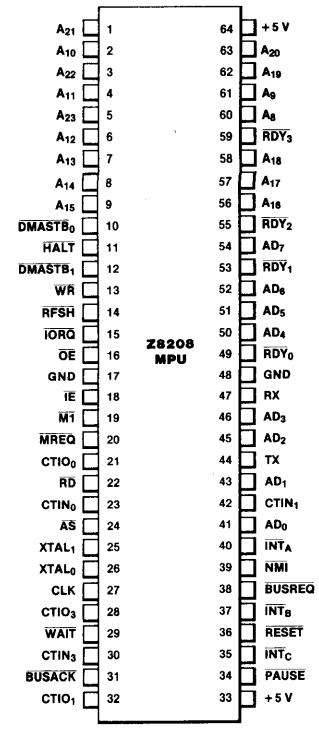


Figure 62. Z8208 Pin Assignments

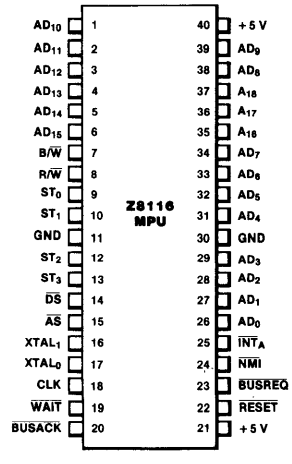


Figure 63. Z8116 Pin Assignments

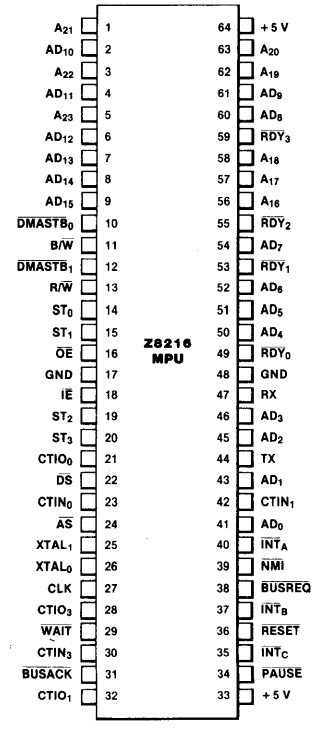


Figure 64. Z8216 Pin Assignments

Z80,000

Family

Zilog

Pioneering the Microworld

Zilog Z80,000™ Family

Provides 16- and 32-Bit Microprocessor Solutions

September 1983

Zilog continues its tradition of state-of-the-art microprocessor components with the introduction of the 32-bit Z80,000 CPU and the Z8070 floating-point Arithmetic Processing Unit (APU). These two devices bring the performance of super minicomputers and mainframe computers into the realm of microprocessor-based systems. The advances in VLSI technology used in these integrated circuits herald a major breakthrough in the range of options available to the systems designer.

The Z80,000. The Z80,000 CPU provides the flexibility of a 16-bit or 32-bit system configuration with the performance of a 32-bit CPU. Oriented to applications in which high throughput is required, its file of 16 general-purpose registers handles bytes, words, and long words with equal facility. The rich instruction set combines powerful addressing modes and operations in a manner that aids assembly-language coding of time-critical applications, and still provides the completeness desirable for efficient compiler-generated code.

The Z80,000 CPU can be configured under software control to use 16-bit logical addresses (ideally suited for high-speed controller applications) or 32-bit addresses (for large-system tasks). The 32-bit address modes support both a linear addressing space and an alternative segmented addressing space, which are selected by the user according to the application's requirements.

Other system features include System and Normal modes of operation, a sophisticated trapping mechanism, a high-performance bus structure, and built-in multiprocessor support. Finally, the device has a high-performance interface to the Z8070 Arithmetic Processing Unit so that the two devices can operate in tandem to execute floating-point instructions in the CPU's instruction stream.

An on-chip cache and a memory management unit (MMU), coupled with a sophisticated instruction pipeline, enable the Z80,000 to execute instructions at a rate of up to one instruction per processor cycle. The 256-byte cache provides an automatic buffering mechanism to hold the most recently fetched instructions and data on the chip itself. Thus, subsequent references to these items do not require lengthy memory transactions but instead can be fetched in a single processor cycle. The memory management unit on the chip contains all the information needed to translate the most recently used logical addresses generated by the CPU into the physical addresses used by the memory system. With each address translation, access attributes are automatically checked to determine whether or not the access is permitted. The MMU can be used to implement a virtual memory or can be disabled entirely for applications that do not need memory management.

Peripheral Support. The Z80,000 uses Zilog's Z-BUS™, so the entire Z8000 family of circuits are available for use with it. Multifunction Z-BUS peripherals are extensively programmable, so each can be precisely tailored to an application. Counting, timing, and parallel I/O are tasks handled by the Z8036 Z-CIO Counter/Timer and Parallel I/O Unit, which has three 16-bit counter/timers and three I/O ports.

Data communications are the domain of the Z8030 Z-SCC Serial Communications Controller and the Z8031 Z-ASCC Asynchronous Serial Communications Controller, both dual-channel multiprotocol components that between them support all popular communication formats.

Direct memory access components are supplied by the Z8016 Z-DTC DMA Transfer Controller, a fast, dual-channel device that supports I/O-to-memory data transfers without CPU intervention. In addition, the Z-BUS versions of the Z800 can be used as I/O processors, with their on-chip DMA channels programmed to transfer data in a Z80,000-based system.

General-purpose control and data-manipulation problems are solved by the Z8090 Z-UPC Universal Peripheral Controller, a complete microcomputer-on-a-chip that uses the Z8 instruction set and features three I/O ports and two 8-bit counter/timers. The Z8038 Z-FIO FIFO Input/Output Interface Unit can be interconnected with asynchronous subsystems of a multiprocessor system to interface any major microprocessor to the Z-BUS. Its buffer depth can be expanded using the Z8060 Z-FIFO Buffer Unit. Other support peripheral circuits that can be used with the Z80,000 are the Z8065 Z-BEP Burst Error Processor and the Z8068 Z-DCP Data Ciphering Processor.

The Z8581 CGC Clock Generator Controller can be used to generate the clock timing required by the Z80,000. This device uses the same technology as the Z80,000 and provides a power-on reset signal and auxilary clocking signals.

Finally, the Z8070 APU Arithmetic Processing Unit provides the floating-point processing power for the Z80,000 CPU.

Z8070 Arithmetic Processing Unit. The Z8070 Arithmetic Processing Unit (APU) provides high-performance binary floating-point capability for the Z800, Z8000, and Z80,000 CPUs. These processors have built-in Extended Processor Architecture, which enables the CPU and APU to function together to execute floating-point instructions. In each case, the CPU fetches the instruction and controls the data movement in the bus, while the APU interprets the instruction and performs the indicated operation. Thus, the programmer generates one stream of instructions and is unconcerned with the mechanism for sharing the task of executing the floating-point instructions in that stream.

The Z8070 can also be used as an I/O device for other popular microprocessors. In this mode, the device is accessed using I/O instructions to write the instruction command to the Z8070 and to move data to and from the APU. This mode of operation is also useful when the Z8070 is used in a bit-sliced CPU implementation to provide floating-point capability.

The Z8070 follows the proposed IEEE P754 standard in respect to data formats, arithmetic operations, and trap handling. In addition to single, double, and double-extended floating-point formats, the Z8070 handles 32- and 64-bit integers and byte strings of BCD digits. Add, subtract, multiply, and divide operations are supplemented with square root, remainder, and a rich array of comparison instructions. Finally, eight trapping conditions are monitored and trigger the device to either generate an interrupt request to the CPU or handle the trap with the pre-programmed, default trap handler on the chip.

Z8070 Z-APU
Arithmetic Processing Unit

Zilog

Preliminary Product Specification

September 1983

FEATURES

- Fast and complete implementation of proposed IEEE Standard P754 Draft 10.0 for Binary Floating-Point Arithmetic. Performs a single-precision multiplication in under three microseconds (with a 10 MHz clock).
- Data types supported are: Single, Double, and Double Extended floating-point; 16- and 32-bit integer; BCD strings.
- Operations supported include add, subtract, multiply, divide, square root, remainder, and compare.
- Interfaces as coprocessor to Z800™, Z8000™, and Z80,000™ CPUs.
- Speed versions offered from 10 MHz to 25 MHz.
- Provides for conversion of binary integer and Binary Coded Decimal formats to and from floating-point format.
- Can be interfaced through Zilog's Extended Processing Architecture or a general-purpose interface.
- Frees CPU for performance of other tasks.

GENERAL DESCRIPTION

The Z8070 Arithmetic Processing Unit (APU) is an Extended Processing Unit (EPU) designed to perform floating-point arithmetic functions while operating in parallel with a CPU. By monitoring the same instruction stream as the CPU, it is able to identify and execute those instructions intended for it, thereby freeing the CPU to perform other activities (Figure 1).

The APU can use Zilog's Extended Processing Architecture (EPA) for the Z800, Z8000, and Z80000, or it can be integrated into systems based on other popular microprocessors, using a general-purpose interface.

The APU supports several data formats, enabling it to handle a wide range of business and scientific applications. These include three binary floating-point formats and four integer formats, including one for variable length Binary Coded Decimal (BCD) strings. All of the APU's internal numeric manipulations use an 80-bit floating-point format; however, transfers of data between the APU's data registers and CPU registers or memory can use any of the formats desired, as specified in the floating-point instruction.

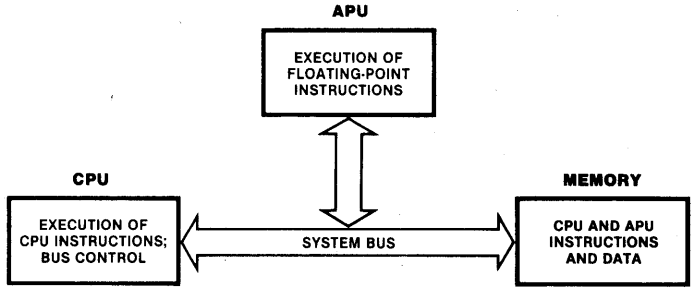


Figure 1. The APU Environment

Floating-point arithmetic operations are performed according to the requirements of the proposed IEEE Standard for floating-point arithmetic. The Z8070 supports:

- Single (32-bit), Double (64-bit), and Extended (80-bit) Precision floating-point number formats.
- Addition, subtraction, multiplication, division, square-root, remainder, and compare operations.
- Conversions between different floating-point formats.
- Conversions between binary integers and floating-point numbers.
- Conversions between decimal integers and floating-point numbers.
- Non-numbers (NaNs) and infinity arithmetic.
- Floating-point exceptions and their handling.

ARCHITECTURE

Overview

The Z8070's contribution to a system is best understood by examining its structure. Internally, the Z8070 is organized as two processors: an Interface Processor and a Data Processor. The two processors have separate clocks, freeing the Data Processor from interface speed constraints. Figure 2 is a block diagram of the Z8070 APU.

The Interface Processor fetches and aligns instructions and data, maintains the internal instruction queue, and executes certain control and data movement instructions independently of the Data Processor. By monitoring CPU status and control signals, the Interface Processor knows when an instruction fetch is to occur and will watch for an Extended Instruction template. It will read and align the instruction and data when the Extended Instruction template has the correct ID number. The user may access the status and control registers of the Interface Processor.

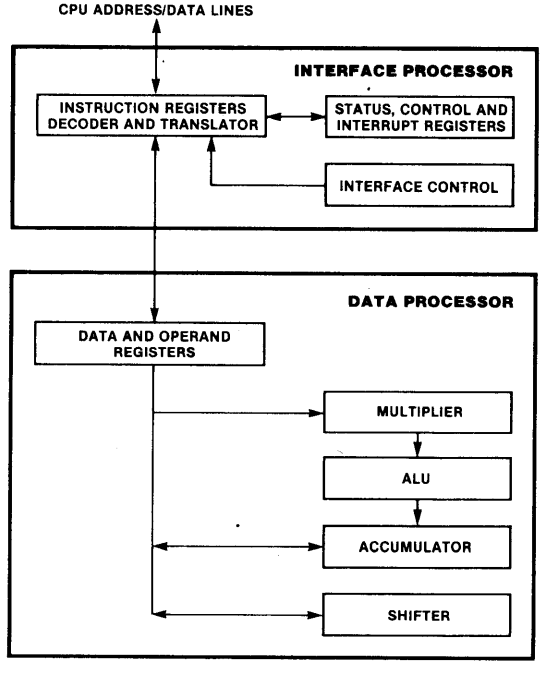


Figure 2. Z8070 Block Diagram

The Data Processor, which operates independently of the Interface Processor, contains eight 80-bit data registers accessible to the user. It also contains the multiplier array, ALU, accumulator, shifter, and temporary registers required for floating-point processing.

The only parts of the Data Processor visible to the user are the eight 80-bit data registers, specified in floating-point instructions as source and/or destination registers, and the two operand registers.

Register Organization

There are eight 80-bit data registers, two 80-bit operand registers, three 32-bit status registers, and one 32-bit and one 16-bit control register in the Z8070. All are accessible to users with the exception of the System Configuration register, which is reserved for privileged users. Figure 3 illustrates the Z8070 register set.

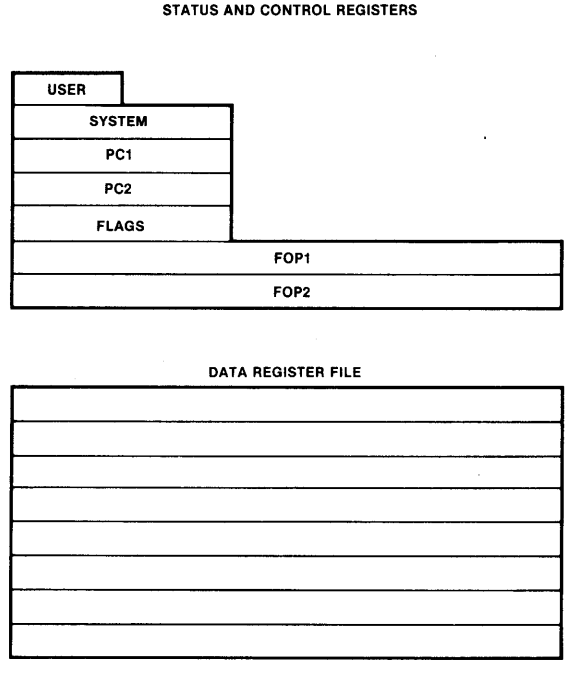


Figure 3. Z8070 Registers

Data Registers. The Z8070 has a data register file of eight 80-bit registers labeled FR0 to FR7 (Figure 4).

Status Registers. There are three 32-bit status registers: the Program Counter registers (PC1 and PC2) and the Flag register.

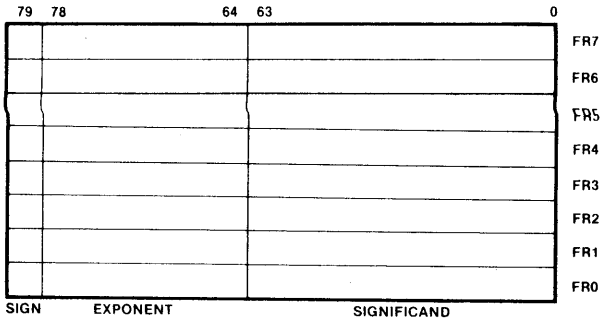


Figure 4. Z8070 Data Register File

Program Counter Registers. PC1 holds the address of the instruction being executed in the Data Processor or the address of any control instruction being executed. PC2 holds the address of any queued instruction (Figure 5).

Flags Register. The Flags register (Figure 6) contains historical information on Z8070 operations as described below.

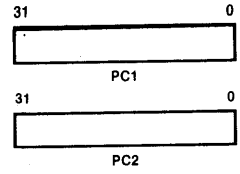


Figure 5. Program Counter Registers

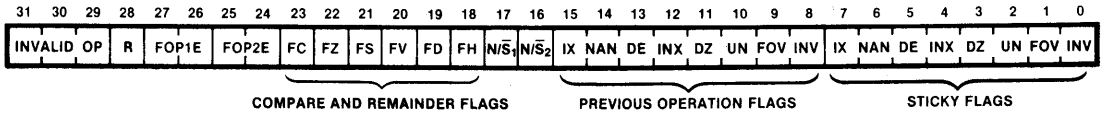


Figure 6. Flags Register

Sticky Flags (0-7). Eight flags are set when the corresponding arithmetic exception occurs, and remain set until they are cleared by the programmer. These flags are:

INV (Invalid Operation)—Indicates an invalid operation or result has occurred (e.g., $\sqrt{3}$).

FOV (Overflow)—Indicates that the absolute value of a floating-point number is too large to be accommodated by the destination format.

UN (Underflow)—Occurs when the absolute value of a number is too small for the destination format, and further denormalization would cause a loss of accuracy.

DZ (Divide by Zero)—Indicates the division of a non-zero finite number by zero.

INX (Inexact Result)—Indicates when the result is inexact due to rounding or an untrapped overflow.

DE (Denormalized number)—Indicates that an operation was performed on a denormalized number.

NAN (Signaling NaN)—Occurs when a Signaling NaN is encountered. (NaN stands for "Not-a-Number", and may be used to force a trap or hold other information.)

IX (Integer Exception)—Occurs when the floating-point number is too large in magnitude to convert to an integer or BCD string, or when an attempt is made to convert a NaN to an integer.

Previous Operation Flags (8-15). The same as the sticky flags described above, but they reflect the exceptions of the previous arithmetic operation.

$N/\overline{S}_1$, $N/\overline{S}_2$ (16-17). Set to indicate Normal mode, and cleared to indicate System mode for PC1 and PC2, respectively.

Compare and Remainder Flags (18-23). Set with comparisons as shown in Table 1.

Table 1. Comparison Results

	<	=	>	Unordered
FC	1	0	0	0
FZ	0	1	0	0
FS	1	0	0	0
FV	0	0	0	1
FD	1	0	1	0

FOP2E (24-25). Contains the two most significant bits of the exponent of operand register FOP2 for use in an overflow exception.

FOP1E (26-27). Contains the two most significant bits of the exponent of operand register FOP1 for use in an overflow exception.

R (28). Rounding bit; 1 if most recent result was rounded up.

Invalid Op (29-31). Contains a code describing the reason for an invalid operation result as follows:

000 Magnitude subtraction of infinities
001 Zero multiplied by infinity
010 Zero divided by zero, or infinity divided by infinity
011 All invalid remainders
100 Unordered compare
101 Square root of a negative number
110 Non-decimal digit on BCD convert

System Configuration Register. The System Configuration register is a 32-bit control register (Figure 7). In systems that distinguish between System and User modes of operation, it is restricted to privileged users.

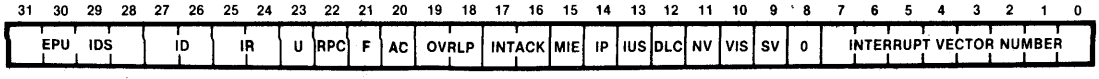


Figure 7. System Configuration Register

Interrupt Vector (0-7). This field identifies the source and cause of an interrupt.

SV (9). Set to shift the interrupt vector left one bit, and set the LSB to zero.

VIS (10). Set when the interrupt vector is to include status information.

NV (11). Set when there is no interrupt vector (leaves lines 3-stated).

DLC (12). Set to disable interrupts from lower priority devices on the interrupt daisy chain.

IUS (13). Set when the interrupt is under service.

IP (14). Set to indicate a pending interrupt.

MIE (15). Set to enable interrupts.

INTACK (16-17). Set to indicate which type of interrupt acknowledge to respond to as follows:

0x = Nonmaskable
10 = Nonvectored
11 = Vectored

OVRLP (18-19). Indicates the Overlap mode as follows:

0x = No Overlap
10 = Intermediate Overlap
11 = Maximum Overlap

AC (20). Set to synchronize processors if CLK.I (interface clock) and CLK.D (data processor clock) are running at different speeds.

F (21). Set if an interrupt service routine will be unable to successfully return to the interrupted program. This happens when two or more floating-point instructions have been fetched after the EPU instruction causing the interrupt, but before the interrupt acknowledge.

RPC (22). Set if an interrupt service routine will need to alter its return address to successfully continue the interrupted program.

U (23). Set when the Z8070 is used.

IR (24-25). Set to indicate the reason for an interrupt as follows:

00 = Arithmetic
01 = Invalid opcode
10 = Invalid EPU ID
11 = Privileged mode violation

ID (26-27). These bits hold the ID of the Z8070 expressed in binary form and are set on power-up with the ID pins. Instructions are executed only if the ID in the opcode matches these bits.

EPUID (28-31). This field contains four bits, one for each possible EPU ID. An instruction specifying an ID whose corresponding bit is a 1 will cause an Invalid EPU ID interrupt.

User Control Register. The User Control register (Figure 8) is a 16-bit register, accessible to all users. The user controls rounding modes and enables and disables traps with this register.

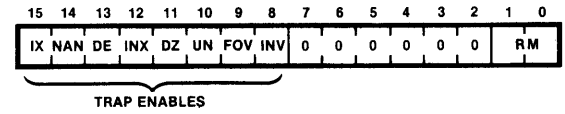


Figure 8. User Control Register

RM (0-1). Sets the rounding modes as follows:

00 = Round to Nearest
01 = Round toward Zero
10 = Round toward Positive Infinity
11 = Round toward Negative Infinity

Trap Enables (8-15). The setting of these bits enables the trap associated with each exception listed below.

INV (Invalid)
FOV (Overflow)
UN (Underflow)
DZ (Divide-by-Zero)
INX (Inexact result)
DE (Denormalized number)
NAN (Signaling NaN)
IX (Integer Exception)

Floating Operand Registers. The Z8070 contains two 80-bit Floating Operand registers (Figure 9), labeled FOP1 and FOP2, which contain the input operand (FOP1) and the default result (FOP2) for use by trap handlers.

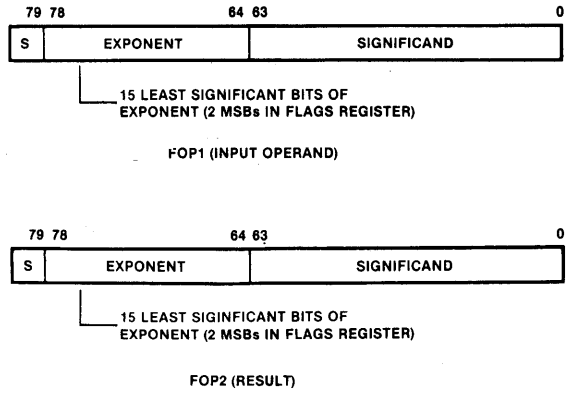


Figure 9. Floating Operand Registers

PROGRAMMING

Floating-point instructions are contained in the same program as standard CPU instructions. To the programmer, instruction execution appears linear, as if a single processor is executing all the instructions. In many cases however, CPU and Z8070 processing can occur in parallel, greatly increasing system throughput.

Parallel processing depends upon the type of APU instruction being executed and the Overlap mode in effect. If the APU receives an instruction involving a data transfer out of the APU, it halts the CPU (asserts $\overline{BSY}$) regardless of the Overlap mode in effect. When No-Overlap mode is in effect, the APU asserts $\overline{BSY}$ whenever any floating-point instruction is received. Figure 10 illustrates instruction processing with and without overlap.

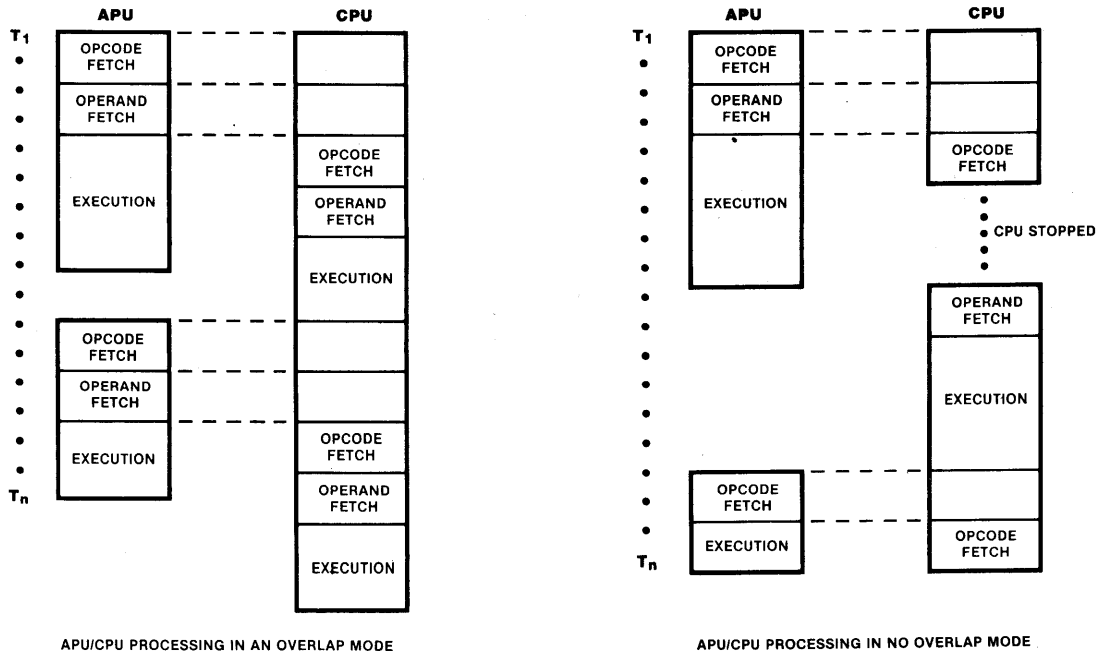


Figure 10. Instruction Processing

In general, the interaction of CPU and APU is transparent to the user. It is possible, however, to arrange programs to take advantage of the parallel processing capabilities inherent in the system.

An interleaving of CPU and Z8070 instructions enhances the ability of a system to process in parallel. Without interleaving, floating-point instructions might be received faster than they can be processed, forcing the Z8070 to halt further CPU processing until the current extended instruction is completed. Also, some instructions take a relatively long time to process (e.g., FSQR, FDIV); interleaving allows the CPU to process instructions while these extended instructions are being processed by the APU.

Parallel processing is facilitated by interleaving instructions as in

```
FADD  F1 @92     !APU instruction!
INC   R2 4       !CPU instruction!
FLD   @R4 F1     !APU instruction!
```

since the increment of R2 can occur while the floating point add is finishing.

Programming constructions like the following

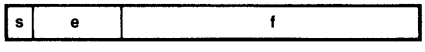
```
FADD  F1 @R2
FLD   @R4 F1
```

cause the Z8070 to halt the CPU to ensure that valid results are read from F1 during the subsequent store operation.

Data Types and Formats

This section describes the different data types and formats that the APU is able to manipulate. These data types include binary floating-point and binary and decimal integers, and can be represented in 32-, 64-, and 80-bit formats.

Binary Floating-Point. All binary floating-point numbers assume the following format:



where the S bit is the Sign bit and specifies a positive (cleared to 0) or negative (set to 1) number. The negative or positive floating-point number is equal to:

$$\text{Significand} \times 2^{(\text{exponent-bias})}$$

The significand portion contains the fraction and the integer bit (in Single and Double Precision binary, the integer bit is implicit). The significand then, is the integer bit followed by the binary point and the fraction. The exponent locates the actual binary point, and the sign bit specifies a positive or negative number.

In the following description of the binary floating-point formats, "s" is the sign, "e" is the exponent, "f" is the fractional part of the significand, and "j" is the integer part (possibly implicit) of the significand.

The value (v) of the 32-bit Single Precision Binary format is determined as follows:

- If $e = 255$ and $f \neq 0$, then v = NaN.
- If $e = 255$ and $f = 0$, then $v = (-1)^s$(infinity).
- If $0 < e < 255$, then $v = (-1)^s 2^{e-127}(1.f)$.
- If $e = 0$ and $f \neq 0$, then $v = (-1)^s 2^{e-126}(0.f)$.
- If $e = 0$ and $f = 0$, then $v = (-1)^s 0$,(zero).

The value of the 64-bit Double Precision binary format is determined as follows:

- If $e = 2047$ and $f \neq 0$, then v = NaN.
- If $e = 2047$ and $f = 0$, then $v = (-1)^s$(infinity).
- If $0 < e < 2047$ then $v = (-1)^s 2^{e-1023}(1.f)$.
- If $e = 0$ and $f \neq 0$, then $v = (-1)^s 2^{e-1022}(0.f)$.
- If $e = 0$ and $f = 0$, then $v = (-1)^s 0$,(zero).

For the 80-bit Double Extended Precision Binary format, the value is determined as follows:

- If $e = 32{,}767$ and $f \neq 0$, then v = NaN.
- If $e = 32{,}767$ and $f = 0$, then $v = (-1)^s$(infinity).
- If $0 < e < 32{,}767$ then $v = (-1)^s 2^{(e-16{,}383)}(j.f)$.
- If $e = 0$ and $j = f = 0$, then $v = (-1)^s 0$, (normal zero).
- If $e = 0$ and j or f is nonzero, then $v = (-1)^s 2^{e-16{,}383}(j.f)$.

The exponent is always biased to ensure a positive value for the purpose of comparisons. Numbers of the same format may then be compared bit by bit from left to right, the first difference determining the ordering. The biases for the floating-point formats are shown in Table 2.

Table 2. Exponent Biases

Format	Exponent Bias
Single	127
Double	1023
Extended	16,383

An exponent of all ones indicates an infinity if the fraction equals zero, or a NaN if the significand is not zero. A Signaling NaN is indicated by the most significant bit (MSB) of the fraction field being zero, and a Quiet (non-trapping) NaN is indicated by the MSB of the fraction being one (in Extended format, Quiet vs. Signaling is determined by bit 62). Single, Double, and Extended formats are shown in Figure 11.

Integers. Integer formats are automatically converted to the 80-bit binary floating-point format when they are loaded into the APU (instructions FLDIL, FLDIQ, and FLDBCD).

Decimal Integers. The Decimal Integer format is one to ten bytes, which includes up to 19 Binary Coded Decimal (BCD) digits and a Sign bit. The Decimal Integer format is illustrated in Figure 12.

Binary Integers. The Long Word and Quad Word Integer formats are shown in Figure 13. These are the only formats which express negative numbers in two's complement form.

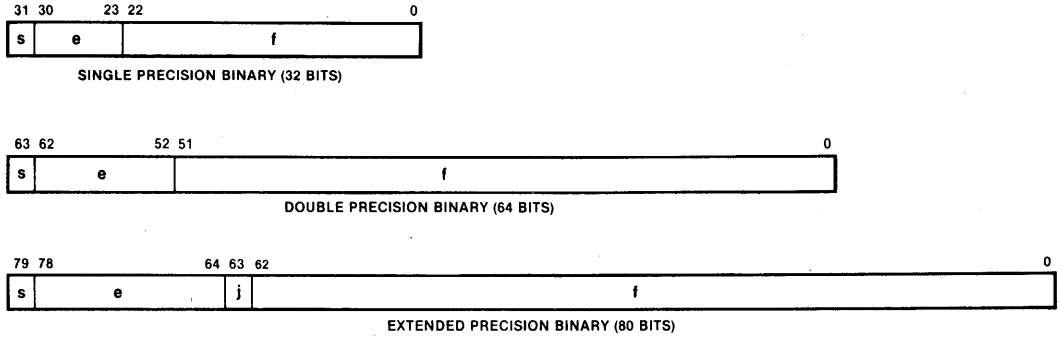


Figure 11. Binary Floating-Point Formats

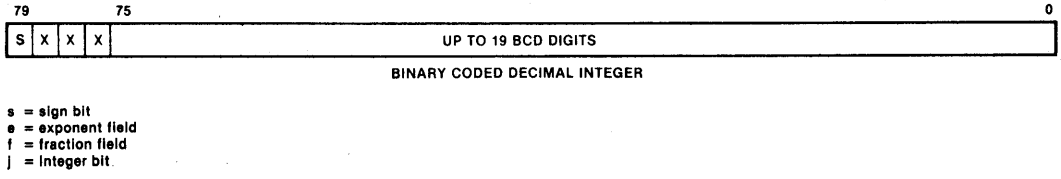


Figure 12. Binary Coded Decimal (BCD) Integer Format

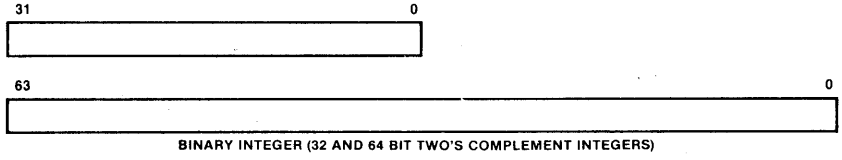


Figure 13. Long-Word and Quad-Word Integer Formats

Addressing Modes

Operands are specified in a floating-point instruction with the addressing modes for each CPU option as shown in Table 3.

Table 3. Addressing Modes

Addressing Mode	Universal	Z800	Z8000	Z80,000
Z8070 Register	F	F	F	F
CPU Register	CPU	R	R	R
Indirect Register	M	IR	IR	IR
Direct Address	M	DA	DA	DA
Index	M	X	X	X
Immediate	M			IM
Relative	M	RA		
Stack Pointer Relative	M	S		
Base Index	M	BX		

F = APU register
CPU = CPU register
M = memory

Assembler Syntax

Floating-point instructions are of the form:

FXXX[S,D] rnd dst,src

The opcode suffix [Single (S), Double (D), or Extended (no suffix)], refers to the size of the source operand. "rnd" refers to the precision to which the result of the operation is rounded. SGL is single precision, DBL is double precision, and no suffix is extended precision.

Instruction Set

The floating-point instruction set provides the following types of instructions:

- Primary arithmetic operations
- Load and store operations
- Compare operations
- Secondary arithmetic operations
- Control operations

Primary Arithmetic Operations

Mnemonic	Operands	Addressing Modes	Operation
FADD **FADDS** **FADDD**	dst,src	src: F,CPU,M dst: F	**Floating Add** dst ← dst + src
FDIV **FDIVS** **FDIVD**	dst,src	src: F,CPU,M dst: F	**Floating Divide** dst ← dst/src
FMUL **FMULS** **FMULD**	dst,src	src: F,CPU,M dst: F	**Floating Multiply** dst ← dst*src
FREMSTEP	dst,src	src: F,CPU,M dst: F	**Floating Remainder Step** dst ← dst REM src
FREMSTEPF	dst,src	src: F dst: F	**Floating Remainder Step and Transfer Flags to CPU** dst ← dst REM src CPU ← Flags
FSQR **FSQRS** **FSQRD**	dst,src	src: F,CPU,M dst: F	**Floating Square Root** dst ← SQR (src)
FSUB **FSUBS** **FSUBD**	dst,src	src: F,CPU,M dst: F	**Floating Subtract** dst ← dst − src

Load and Store Operations

Mnemonic	Operands	Addressing Modes	Operation
FLD **FLDS** **FLDD**	dst,src	src: F,CPU,M dst: F or src: F dst: CPU,M	**Floating Load** dst ← src
FLDBCD	dst,src	dst: F src: CPU,M or dst: CPU,M src: F	**Floating Load BCD Integer** dst ← Float (BCD-src) dst ← BCD (float-src)
FLDIL	dst,src	dst: F src: CPU,M or dst: CPU,M src: F	**Floating Load Binary Integer Long Word** dst ← Float (src) dst ← Fix (src)
FLDIL	dst,src	dst: F src: CPU,M or dst: CPU,M src: F	**Floating Load Binary Integer Long Word** dst ← Float (src) dst ← Fix (src)
FLDIQ	dst,src	dst: F src: CPU,M or dst: CPU,M sdc: F (note: CPU = 64-bit register)	**Floating Load Binary Integer Quad Word** dst ← Float (src) dst ← Fix (src)
FLDM	dst,src,n (n = 1,2)	dst: F src: CPU,M or dst: CPU,M src: F (note: F,CPU,M = 80- or 160-bit location)	**Floating Load Multiple** dst ← src
FLDTL	dst,src	dst: CPU,M src: F (note: dst is 32 bits)	**Floating Load and Truncate to Integer Long Word** dst ← Int (src)
FLDTQ	dst,src	dst: CPU,M src: F (note: dst is 64 bits)	**Floating Load and Truncate to Integer Quad Word** dst ← Int (src)

Compare Operations

Mnemonic	Operands	Addressing Modes	Operation
FCP **FCPS** **FCPD**	dst,src	dst: F src: F,CPU,M	**Floating Compare** dst – src, set APU flags
FCPF	dst,src	dst: F src: F	**Floating Compare and Transfer *Flags to CPU*** dst – src CPU ← flags
FCPFX	dst,src	dst: F src: F	**Floating Compare, Transfer Flags to CPU, and Raise Exception if Unordered** dst – src CPU ← flags
FCPX **FCPXS** **FCPXD**	dst,src	dst: F src: F,CPU,M	**Floating Compare and Raise Exception if Unordered** dst – src, set APU flags
FCPZ **FCPZS** **FCPZD**	dst,src	dst: F,CPU,M	**Floating Compare with Zero** dst – 0, set APU flags
FCPZF	dst	dst: F	**Floating Compare with 0, and Transfer Flags to CPU** dst – 0 CPU ← flags
FCPZFX	dst	dst: F	**Floating Compare with 0, Transfer Flags to CPU, and Raise Exception if Unordered** dst – 0 CPU ← flags
FCPZX **FCPZXS** **FCPZXD**	dst	dst: F,CPU,M	**Floating Compare with Zero and Raise Exception if Unordered** dst – 0 set APU flags

Secondary Arithmetic Operations

Mnemonic	Operands	Addressing Modes	Operation
FABS **FABSS** **FABSD**	dst,src	dst: F src: F,CPU,M	**Floating Absolute Value** dst ← |src|
FCLR	dst	dst: F	**Floating Clear** dst ← +0
FINT **FINTS** **FINTD**	dst,src	dst: F src: F,CPU,M	**Floating Round to Floating Integer** dst ← Float [Int (src)]
FNEG **FNEGS** **FNEGD**	dst,src	dst: F src: F,CPU,M	**Floating Negation** dst ← (−src)

Control Operations

Mnemonic	Operands	Addressing Modes	Operation
FLDCTL	dst,src	dst: FCTL src: CPU,M or dst: CPU,M src: FCTL	**Floating Load Control** dst ← src
FLDCTLB	dst	dst: Fsel	**Floating Load Control Byte** CPU ← flags
FRESFLG	src	dst: FFLAGS src: flaglist	**Floating Reset Flag** FFLAGS (flaglist) ← 0
FRESTRAP	src	dst: USER src: traplist	**Floating Reset Trap** USER (traplist) ← 0
FSETFLG	src	dst: FFLAGS src: flaglist	**Floating Set Flag** FFLAGS (flaglist) ← 1
FSETMODE	src	dst: FMODE src: modelist	**Floating Set Mode** FMODE ← modelist
FSETTRAP	src	dst: USER src: traplist	**Floating Set Trap** USER (traplist) ← 1

SIGNAL DESCRIPTIONS

The following section describes each pin function of the Z8070 APU. Depending on which CPU option is chosen (pins OPT_0 and OPT_1), some of these functions do not apply. Figure 14 shows a Z8070 pin-out with pin assignments, and Figure 15 gives the functional pin-out for each of the four possible interface options.

$\overline{\text{ABORT}}$. *Abort (input, active Low).* $\overline{\text{ABORT}}$ is asserted to cause an instruction abort.

AD_0–AD_{31}. *Address/Data (inputs/outputs, active High, 3-state).* Multiplexed address and data lines.

$\overline{\text{AS}}$. *Address Strobe (input, active Low).* The rising edge of $\overline{\text{AS}}$ indicates the beginning of a transaction and shows that the address, status, and control signals are valid.

BL/$\overline{\text{W}}$ and BW/$\overline{\text{L}}$. *Byte, Word, and Long Word (inputs).* These signals specify the data transfer size as follows:

BL/$\overline{\text{W}}$	BW/$\overline{\text{L}}$	Size in bits
1	1	8
0	1	16
1	0	32

$\overline{\text{BRST}}$. *(Burst, input, active Low).* $\overline{\text{BRST}}$ active indicates that the CPU may generate burst transfers.

$\overline{\text{BRSTA}}$. *Burst Acknowledge (input, active Low).* A Low on $\overline{\text{BRSTA}}$ indicates that the memory can support burst transfers.

$\overline{\text{BSY}}$. *Busy (output, active Low).* The $\overline{\text{BSY}}$ signal is used by the Z8070 to halt the CPU during stores, and also to implement overlap functions. It is associated with the following CPU signals:

CPU	Signal
Z800	$\overline{\text{PAUSE}}$
Z8000	$\overline{\text{STOP}}$
Z80,000	$\overline{\text{EPUBSY}}$

$\overline{\text{BUSACK}}$. *Bus Acknowledge (input, active Low).* When $\overline{\text{BUSACK}}$ goes Low, the EPU 3-states the AD lines until it goes High again and an EPU-to-memory or CPU transfer is required.

CLK.D. *Data Processor Clock (input).* CLK.D is provided by the system and runs the data processor portion of the Z8070.

CLK.I. *Interface Processor Clock (input).* Interface Processor clock of the Z8070.

$\overline{\text{CS}}$. *Chip Select (input, active Low).* $\overline{\text{CS}}$ signals the beginning of a Z8070 transaction and is valid only for the length of a single transaction or machine cycle.

DB_0–DB_n. *Data Bus (input/outputs, active High, 3-state).* These are the data lines for the Universal interface, where n = 7, 15 or 31.

$\overline{\text{DS}}$. *Data Strobe (input, active Low).* $\overline{\text{DS}}$ provides timing for data transfers on the bus.

ID_0–ID_1. *ID Select (inputs, active High).* These signals establish the EPU ID during reset.

IEI. *Interrupt Enable In (input, active High).* IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt-driven device sharing a common interrupt request line to the CPU. A High IEI indicates that no other higher priority device has an interrupt under service.

IEO. *Interrupt Enable Out (output, active High).* IEO is High only if IEI is High and the CPU is not servicing a Z8070 interrupt and the Z8070 is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices. IEO is tied high for the last device on the chain.

$\overline{\text{INT}}$. *Interrupt Request (output, open-drain, active Low).* This signal is activated when the Z8070 requests an interrupt.

N/$\overline{\text{S}}$. *Normal/System mode (input).* High for Normal mode, Low for System mode.

OPT_0–OPT_1. *CPU Option (input, active High).* These signals establish the CPU option during reset as follows:

OPT_0	OPT_1	CPU Interface
0	0	Universal
0	1	Z80,000
1.	0	Z800
1	1	Z8000

$\overline{\text{RESET}}$. *Reset (input, active Low).* When asserted, $\overline{\text{RESET}}$ forces a hardware reset to power-on condition.

RSP_0–RSP_1. *Response (input, active High).* These lines encode the response monitored by the APU to transactions initiated by the Z80,000 CPU as follows:

RSP_0	RSP_1	
0	1	Bus Error
1	0	Bus Retry
0	0	Wait
1	1	Ready

R/$\overline{\text{W}}$. *Read/Write (input, Low = Write).* Indicates whether the CPU is performing a read or write operation.

$\overline{\text{SIP}}$. *Sequence in Progress (output, active Low).* This signal is asserted by the APU and held Low until the instruction and any associated data transfer is completed.

SN_0–SN_6. *Segment Number (input, active High).* These lines contain the segment number portion of a memory address.

ST_0–ST_3. *Status (input, active High).* These lines specify the type of bus transaction as described in the appropriate CPU Technical Manual.

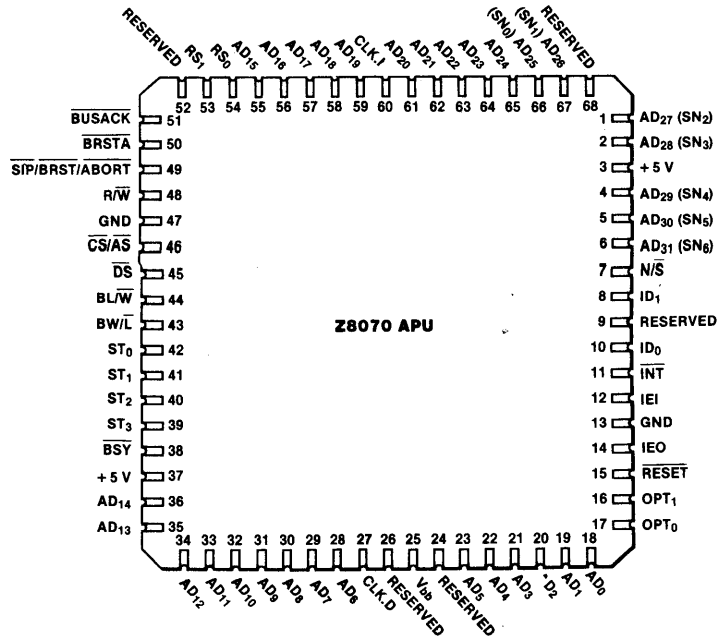


Figure 14. Pin Assignments

EXTERNAL INTERFACE

The four different interfaces for the Z8070 are the Z800, Z8000, Z80000, and Universal interface. The Z800, Z8000, and Z80000 are Z-BUS® type interfaces with the CPU and APU fully interlocked in hardware. No software polling or wait instructions are required to prevent overrunning of the Z8070's instruction queue with the Z-BUS interfaces. Depending on the type of system, the Universal interface may require additional logic to implement the interface. In general, APU-memory transactions are performed with the same bus cycles as CPU-memory transactions. The following sections describe interface specific features of the Z8070.

Interface Types

Z-BUS Interface. An important feature of the Z-BUS CPUs (Z800, Z8000, and Z80000) is their Extended Processing Architecture (EPA). This facility provides a mechanism by which the basic instruction set can be extended via external EPUs such as the Z8070 APU.

The execution of floating-point instructions is controlled by an extended processor enable bit in the CPU. When this bit is zero, it indicates that there is no EPU connected to the CPU, causing the CPU to trap to a software trap handler whenever an extended instruction is encountered. This allows the operation of the extended instruction to be performed by software, and provides the ideal tool for emulating an EPU during development of systems intended to later contain an EPU.

If the extended instruction indicates a transfer of data between the Z8070's internal registers and the main memory, the CPU will calculate the memory address and generate the appropriate timing signals ($\overline{AS}$, $\overline{DS}$, MREQ, etc.), but the data transfer itself is between the Z8070 and memory (over the AD lines). If a transfer of data between the CPU and APU is indicated, the sender places the data on the AD bus while $\overline{DS}$ is active.

If the extended instruction indicates an internal operation to be performed by the Z8070, the Z8070 begins execution of the task and the CPU is free to continue on to the next instruction. Processing then proceeds simultaneously in both the CPU and the Z8070 until a second extended instruction is encountered that is destined for the Z8070.

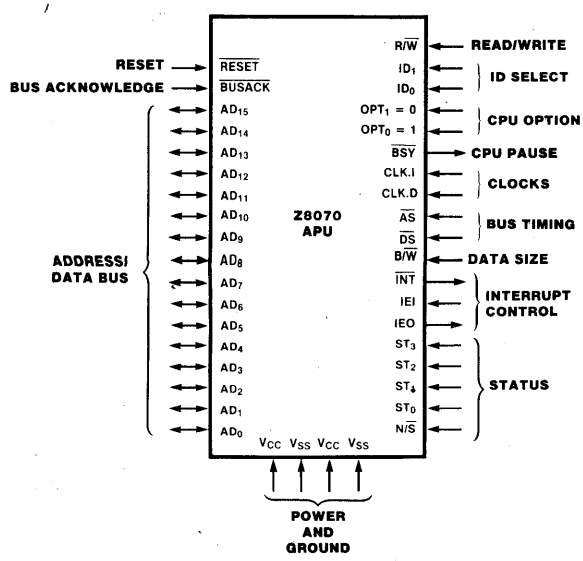


Figure 15a. Z8070/Z800 Interface

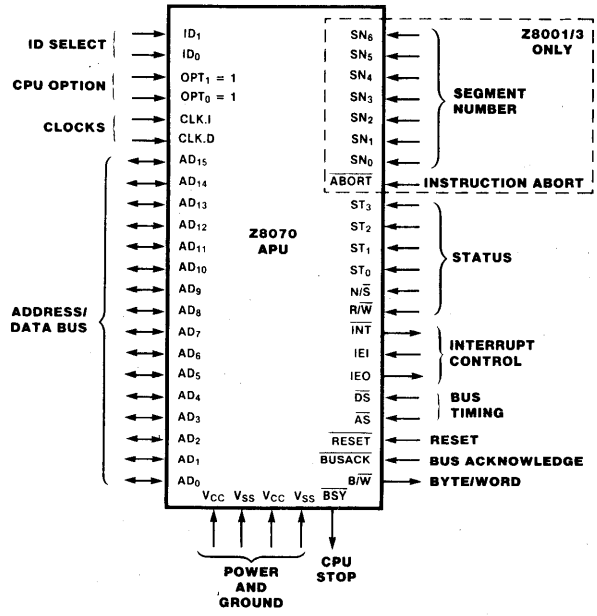


Figure 15b. Z8070/Z8000 Interface

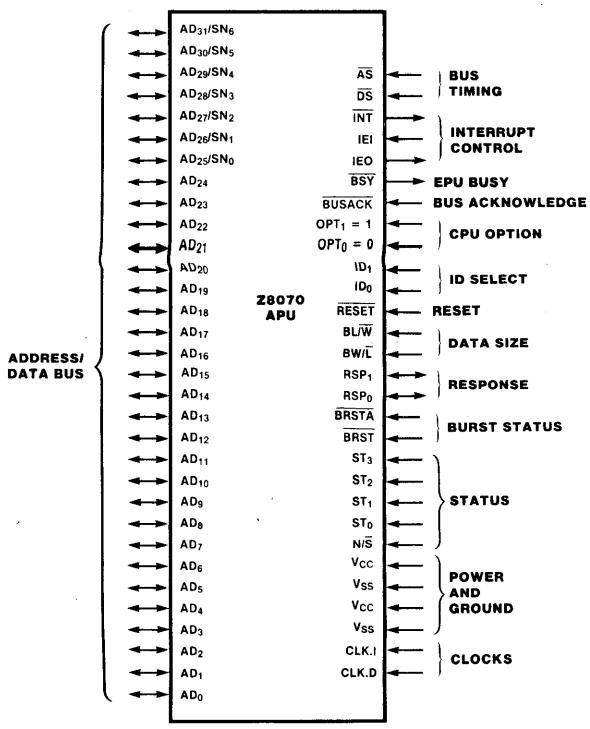


Figure 15c. Z80,000/Z8070 Functional Interface

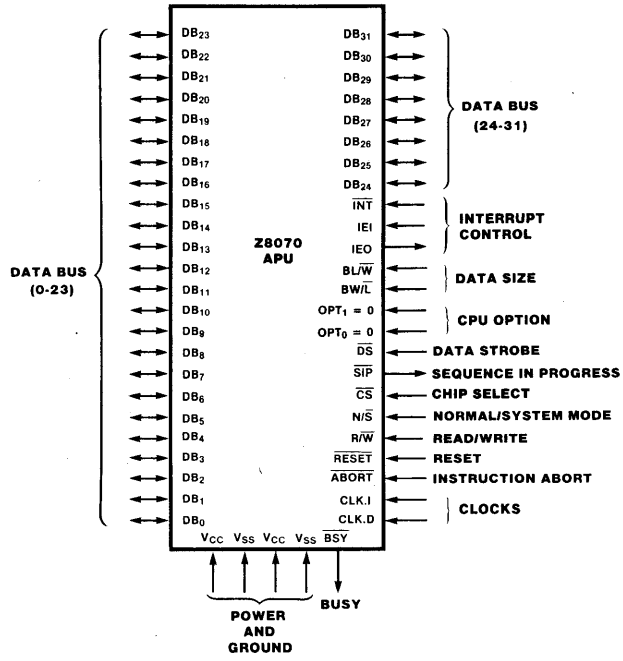


Figure 15d. Z8070 Universal Interface

The Extended Processing Architecture also offers a provision to preclude extended instruction overlapping. The APU connects to the CPU via the $\overline{BSY}$ line so that if the APU is requested to perform a second extended instruction before it has completed the previous one, it can halt the CPU until execution of the previous instruction is complete. $\overline{BSY}$ is also asserted when a store opcode is received by the APU, and it is held active until the requested data is ready for the CPU to transfer.

With the Z800 interface, the CPU indicates an instruction fetch to the APU whenever it places a floating-point instruction on the bus. The Z8070 then translates the instruction and performs the operation, asserting $\overline{BSY}$ if the opcode indicates a store.

For the Z8000 interface, if the EPA bit is set it indicates that an EPU is connected to the CPU. The CPU begins operation by fetching an instruction and determining whether it is a CPU or an EPU command. The EPU monitors the Z-BUS at the same time, looking for an extended instruction template. The CPU performs any address calculations required by the extended instruction and, if the instruction specifies the transfer of data, the CPU generates the timing signals for the transfer.

The Z8070 monitors the activity on the Address/Data (AD) bus. If the instruction fetched is an extended instruction, all EPUs and the CPU latch the instruction. If the Z8070 is not busy when the instruction and data intended for it appear, the floating-point instruction is executed. If the Z8070 is busy, the CPU is stopped until the Z8070 is no longer busy.

The Z80,000 CPU differs slightly from the Z800 and Z8000 CPUs described above. When the Z80,000 CPU detects an extended instruction, it first samples $\overline{BSY}$. If $\overline{BSY}$ is inactive, the CPU signals an EPU transaction (status 0100) and places the extended instruction on the AD bus. If $\overline{BSY}$ is active, the CPU samples $\overline{BSY}$ every bus clock cycle until it is inactive. The CPU may acknowledge bus requests or interrupt requests during this period, and if this should happen before all associated data transfers are complete, the CPU saves the address of the extended instruction.

Universal Interface. With the Universal interface, the Z8070 does not monitor the bus but, rather, waits until its CS line becomes active, indicating that the instruction on the bus is intended for it. The APU then reads data from the bus during each Data Strobe until it has collected the full instruction and associated data. The decoded instruction informs the Z8070 of the type of operation it is to perform. When CS goes Low, it forces the Sequence in Progress signal ($\overline{SIP}$) Low, and $\overline{SIP}$ stays Low until the last bus transaction associated with the instruction is complete.

The Universal interface can be set for a data bus width of 8-, 16-, or 32-bits with the BL/$\overline{W}$ and BW/$\overline{L}$ pins. A set of interrupt control signals ($\overline{INT}$, IEI, IEO) permit the integration of the Z8070 in a daisy-chain priority interrupt scheme.

Bus Transactions

The following section describes bus transactions for the Z800, Z8000, Z80,000, and Universal interfaces.

Z800/Z8070 Bus Transactions. The 16-bit Z800 MPUs, Z8116 and Z8216, incorporate Zilog's Extended Processing Architecture (EPA) for Extended Processing Units. The Z8116 is a 40-pin device that contains an MMU, clock oscillator, and refresh controller on-chip. It does not have a $\overline{PAUSE}$ input. The Z8216 is a 64-pin device which additionally contains four DMA channels, four counter/timers, and a UART, and does have a $\overline{PAUSE}$ input. When the Z800 encounters an EPU instruction, it fetches the following EPU template from memory with instruction fetch status. (The opcode and addressing mode portion of the instruction may be executed from cache, but the template will always be fetched from memory with status 1101 or 1100). When an EPA template with the appropriate ID number is detected, the Z8070 obtains or places data or status information on the bus using the Z800 generated control signals and performs its function as directed.

The Z800 MPU is responsible for instructing the APU and delivering operands and data to it. The Z8070 recognizes templates intended for it and executes them, using data supplied with the template and/or data within its internal registers. There are three classes of APU instructions:

- Data transfers between main memory and APU registers.
- Data transfers from APU registers to the CPU's accumulator.
- APU internal operations.

Six Z800 addressing modes may be utilized with transfers between APU registers and the CPU and main memory:

- Direct Address
- Indirect Register
- Indexed
- Relative
- Stack Pointer Relative
- Base Index

The Z8070 connects to the Z800 via the $\overline{BSY}$ ($\overline{PAUSE}$) signal, so that if the APU is requested to perform a second floating-point instruction before it has completed the previous one, it can stop the CPU until execution of the previous floating-point instruction is complete. $\overline{BSY}$ will also be asserted if the instruction is a store.

Z8070/Z800 instruction execution is illustrated in Figure 16. The Z800 begins operation by fetching an instruction and determining whether or not it is an extended instruction. If it is, the state of the EPU Enable bit in the Trap Control register is examined. If the EPU Enable bit is

zero, The CPU generates a trap and may simulate the APU in software. If the EPU Enable bit is set to 1, the four byte EPA template is fetched from memory, with 1 1 0 0 indicated on status lines ST_0–ST_3. After fetching the template, the Z800 will, if necessary, transfer appropriate data between the CPU and memory or between the CPU and the APU. If the APU is not busy when the data and template for it appear, the template is executed. If the APU is still processing a previous instruction, it asserts $\overline{BSY}$ to halt further execution of CPU instructions until execution is complete. After the execution of the template is complete, the APU releases the $\overline{BSY}$ line and CPU instruction execution continues.

APU to CPU transfer transactions (Figure 16) have the same form as I/O transactions, and thus are four clock cycles long (unless extended by $\overline{WAIT}$). ST_3–ST_0 = 1110, and the CPU output R/$\overline{W}$ indicates the direction of data transfer. A Z8070-memory read is illustrated in Figure 17 and a Z8070-memory write is illustrated in Figure 18.

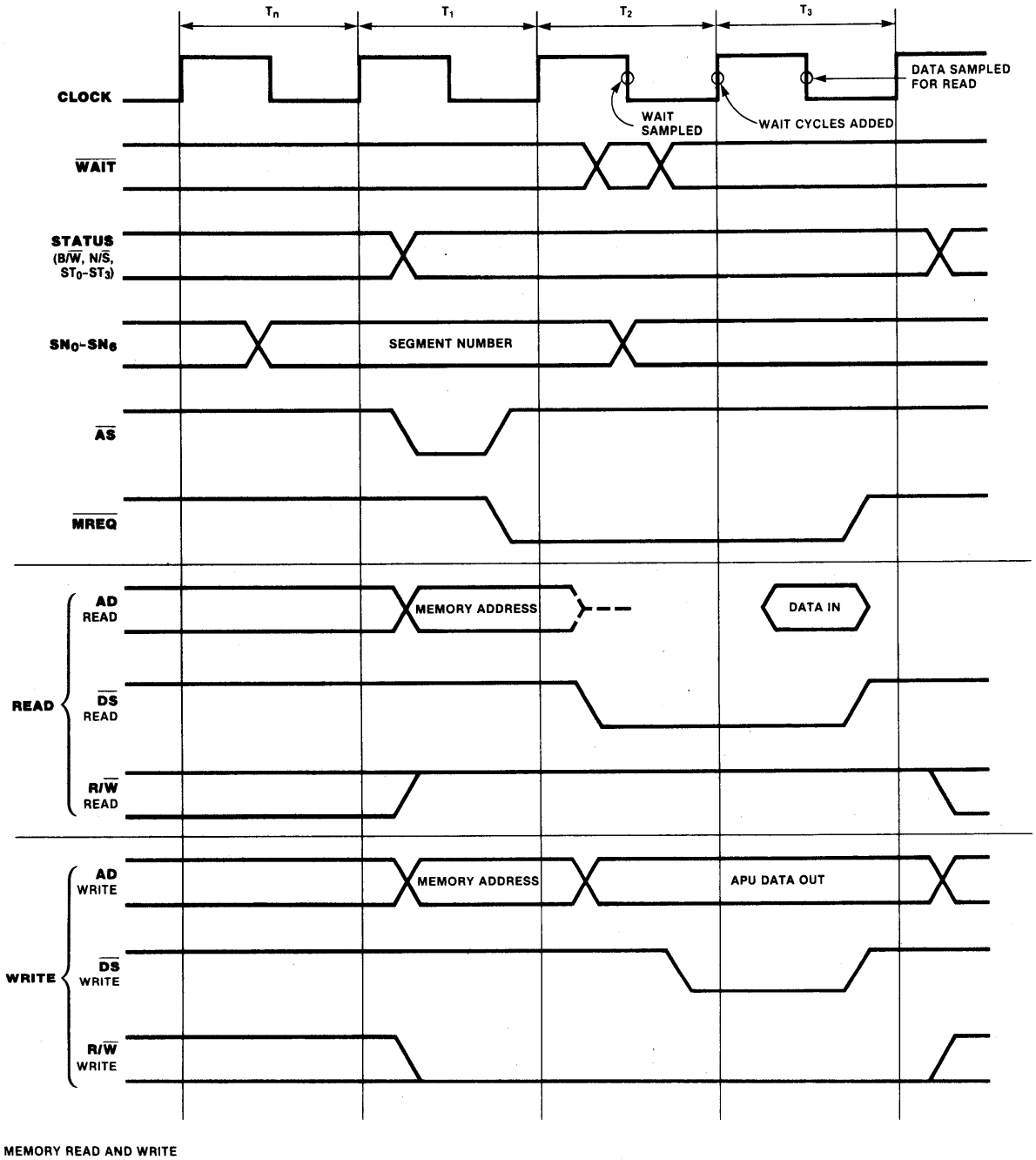


Figure 16. Z8070-Z800 Transfer Transaction

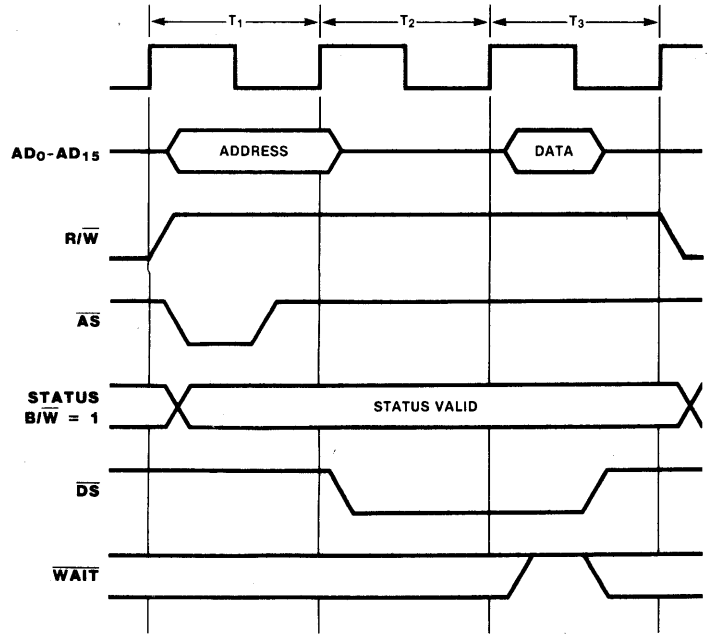


Figure 17. Z8070-Memory Read Transaction (Z800)

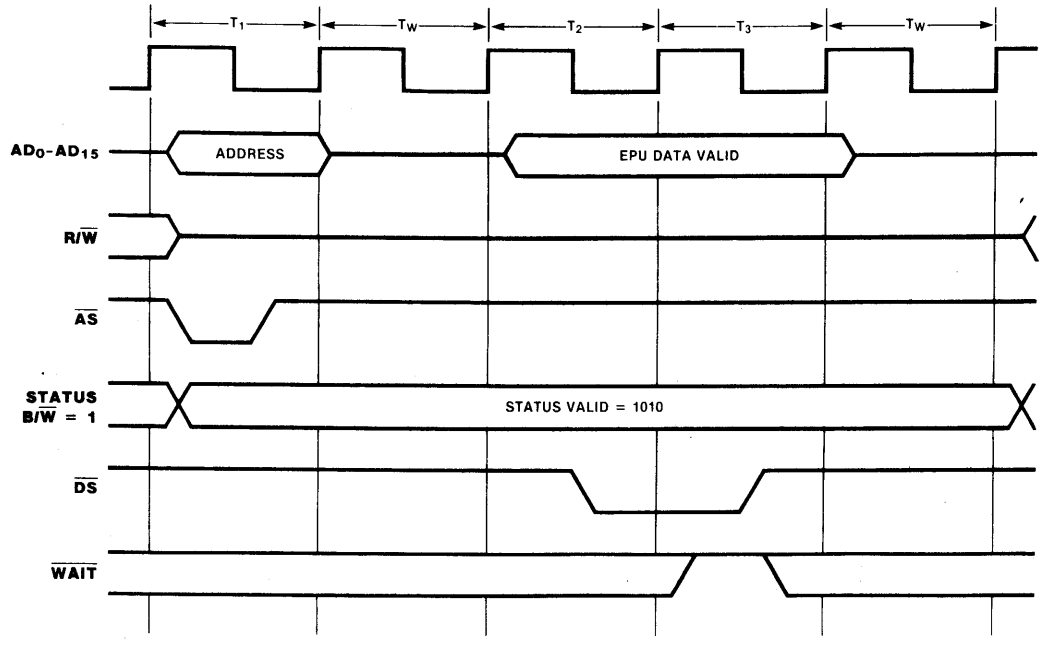


Figure 18. Z8070-Memory Write Transaction (Z800)

Z8000/Z8070 Bus Transactions. Z8000/Z8070 transfer transactions move data between the APU and CPU or between the APU and memory. The CPU can transfer data to or from the APU, and provides address and bus control signals for transfers between the APU and memory.

When the Z8070 is to participate in a memory transaction, the Z8000 places its AD (Address/Data) lines into the high impedance state while $\overline{DS}$ is Low, so that the Z8070 can use them. APU-memory transfer transactions are the same as CPU-memory transactions (Figure 19). The CPU generates the address, and status codes 1010 and 1011 are used. APU-CPU transactions have the same timing relationship as I/O transactions (Figure 20).

In order to know which transaction it is to participate in, the Z8070 tracks the following sequence of events:

1. When the CPU fetches the first word of an instruction (ST_3–ST_0 = 1101), the APU must also capture the instruction returned by memory. If the instruction is an extended instruction, it will have an ID field which indicates whether or not the APU is to execute it.
2. If the instruction is to be executed by the APU, the next non-refresh transaction by the CPU fetches the second word of the instruction (ST_3–ST_0 = 1100). The APU also captures this word.
3. If the instruction involves a read or write to memory, there will be zero or more program fetches by the CPU (ST_3–ST_0 = 1100) to obtain the address portion of the extended instruction. The next 1 to 16 non-refresh transactions by the CPU will transfer data between memory and the APU. The APU supplies the data (R/$\overline{W}$ Low) or captures the data (R/$\overline{W}$ High) for each transaction. In both cases the CPU 3-states its AD lines while data is being transferred ($\overline{DS}$ Low). APU memory transfers are always word oriented (B/$\overline{W}$ Low).
4. If the instruction involves a transfer between the CPU and APU, the next 1 to 16 non-refresh transactions by the CPU transfer data between the APU and CPU (ST_3–ST_0 = 1110).

To follow the above sequence, the Z8070 has to monitor the $\overline{BUSACK}$ line to verify that the transaction on the bus is generated by the CPU. There is no indication on the bus as to which EPU in a multiple EPU system is cooperating with the CPU—this must be determined from the opcodes and ID fields of the extended instruction the APU captures.

With the first two instruction words, the Z8070 determines:

- Whether or not a memory access will be made.
- The number of words of data to be transferred for APU-memory or APU-CPU transfers.
- The operation to be performed on the data.

A final aspect of Z8000/Z8070 interaction is the use of the $\overline{BSY}$ ($\overline{STOP}$) signal. If the system is in Non-Overlap mode, the APU asserts $\overline{BSY}$ to stop the CPU whenever it receives a floating-point instruction and its associated data. If overlapping is allowed, when the Z8070 begins to execute a floating-point instruction the CPU can continue fetching and executing instructions. If the CPU fetches another floating-point instruction before the first one has completed execution, the APU asserts $\overline{BSY}$ until

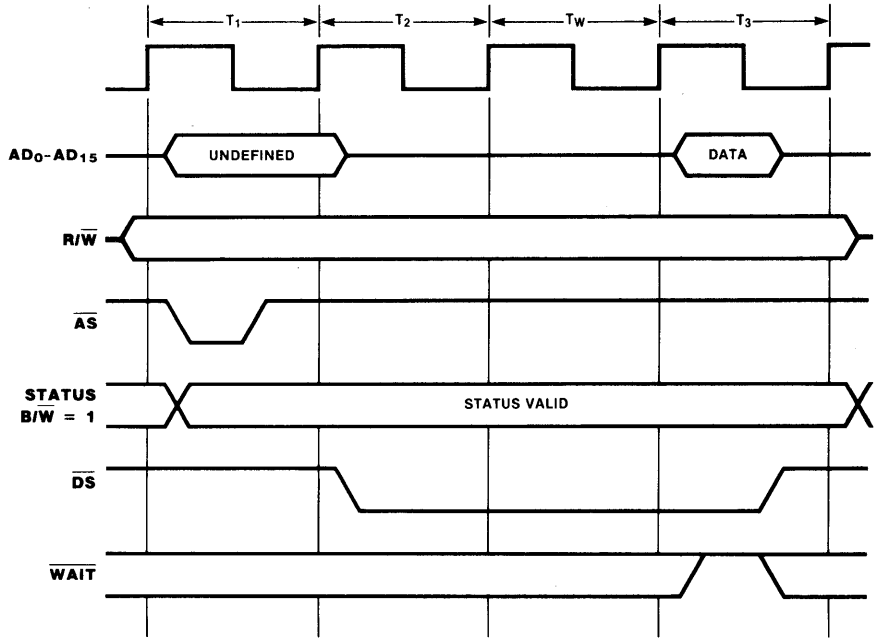


Figure 19. Z8070-Memory Transfer Transaction (Z8000)

execution of the previous floating-point instruction is complete. (In Intermediate Overlap mode $\overline{BSY}$ is not asserted until a third APU instruction is fetched.) $\overline{BSY}$ is always asserted if an instruction involves a store.

Z80,000/Z8070 Bus Transactions. When the Z80,000 CPU encounters an EPA instruction and the EPA bit in the FCW is set to 1, the CPU broadcasts the first two words of the instruction to the EPUs in the system using the CPU-EPU instruction transfer transaction. All EPUs in the system recognize the transaction, but the APU is chosen specifically in bits 16 and 17 of the EPU instruction. The Z80,000 also transfers the PC value for the instruction, which the APU saves for use in exception handling. If data transfers are required to complete the instruction, the Z80,000 controls the data transfer transactions while the Z8070 drives or receives the data.

The signal $\overline{BSY}$, output from the APU and wired to the CPU's $\overline{EPUBSY}$ pin, is used to synchronize the CPU and APU in executing floating-point instructions. The CPU must sample $\overline{BSY}$ inactive before initiating an APU instruction transfer. If data transfers are required, the CPU must sample $\overline{BSY}$ inactive before initiating the transfer.

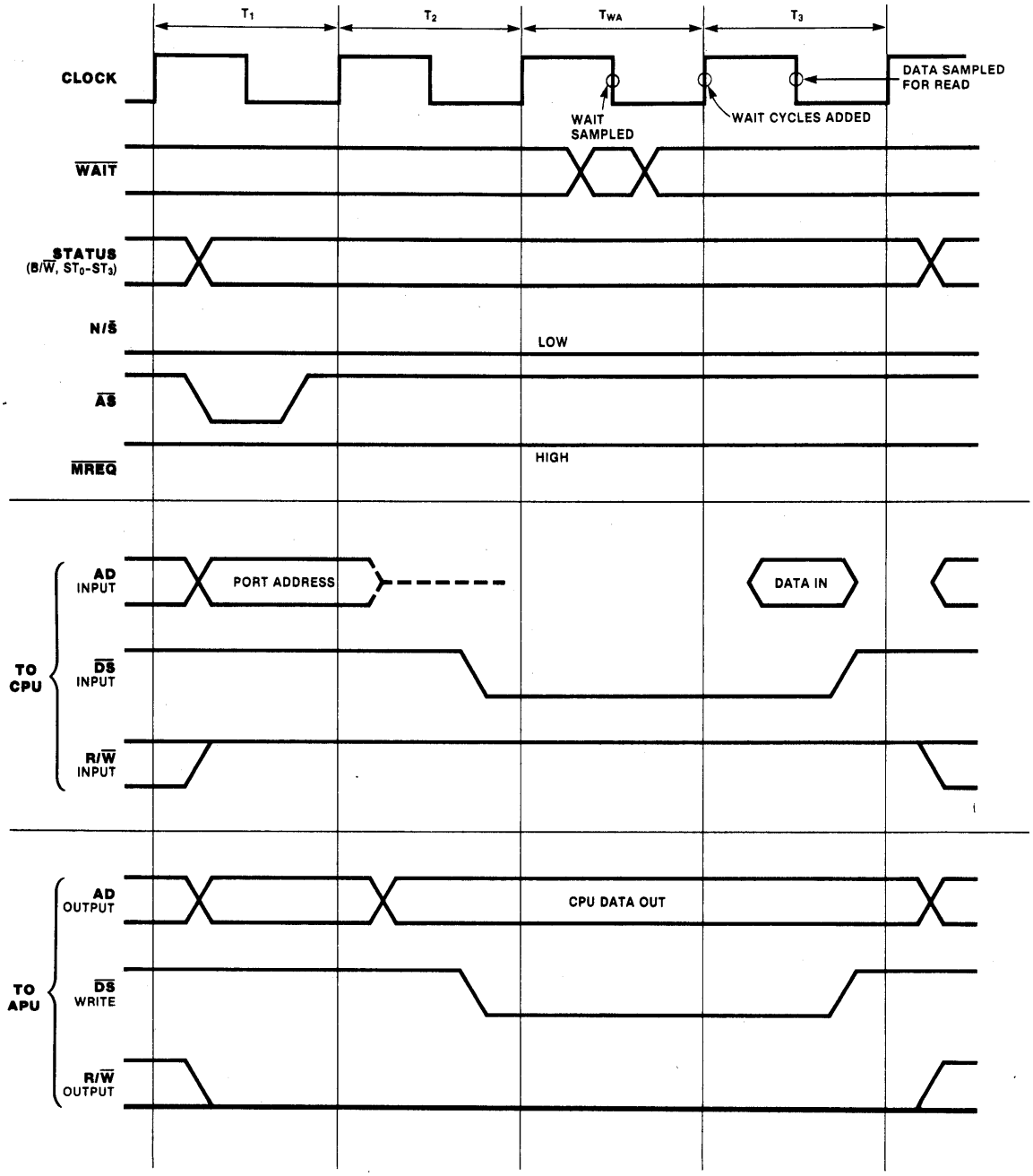
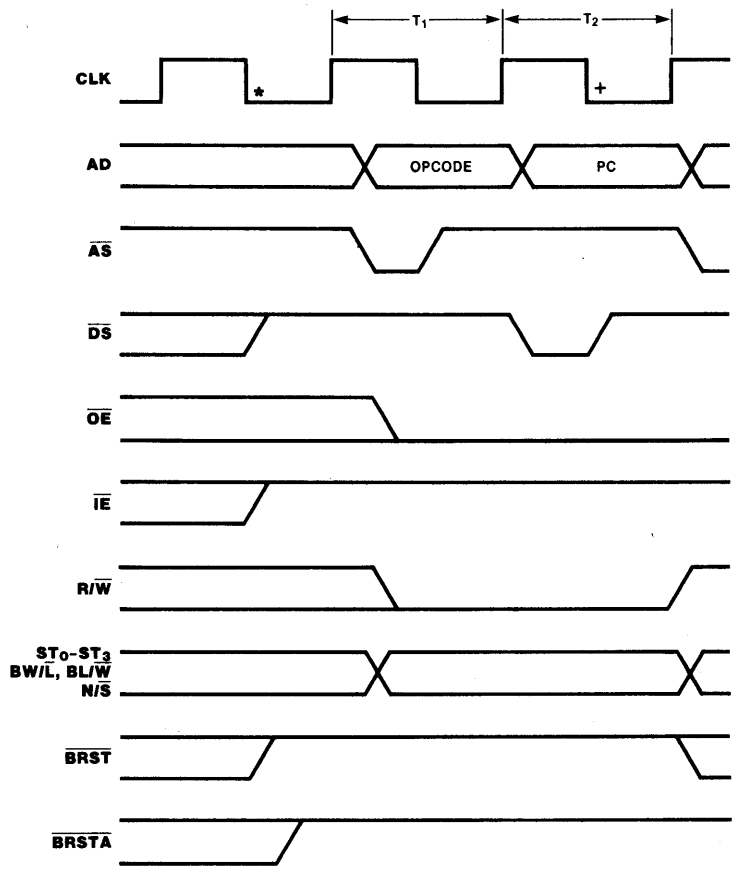


Figure 20. Z8070-Z8000 Transfer Transaction

$\overline{BSY}$ is also used to control the degree of overlap between CPU and APU instruction processing. Ordinarily the CPU can continue processing other instructions after performing the data transfers associated with a floating point instruction, and before the APU has completed executing the instruction. To simplify debugging and recovery from exceptions, a Non-Overlap mode is provided, controlled by the EPU.0 bit in the Z80,000 Hardware Interface Control Register. In Non-Overlap mode, the CPU samples $\overline{BSY}$ in the middle of the bus cycle in which the last data transfer for an extended instruction occurs. If $\overline{BSY}$ is asserted, the Z80,000 ceases processing instructions or interrupts until $\overline{BSY}$ is sampled inactive in the middle of a bus cycle.

CPU-APU Instruction Transfer Transactions. Timing for a CPU-APU instruction transfer transaction, with status 0100, is shown in Figure 21. The transaction begins with Address Strobe to indicate the AD lines and status are valid. During T_1 the AD lines are used to transfer the opcode, which is the first two words of the EPA instruction. At the beginning of T_2 the CPU stops driving the opcode, asserts $\overline{DS}$, and starts driving the PC on the AD lines.

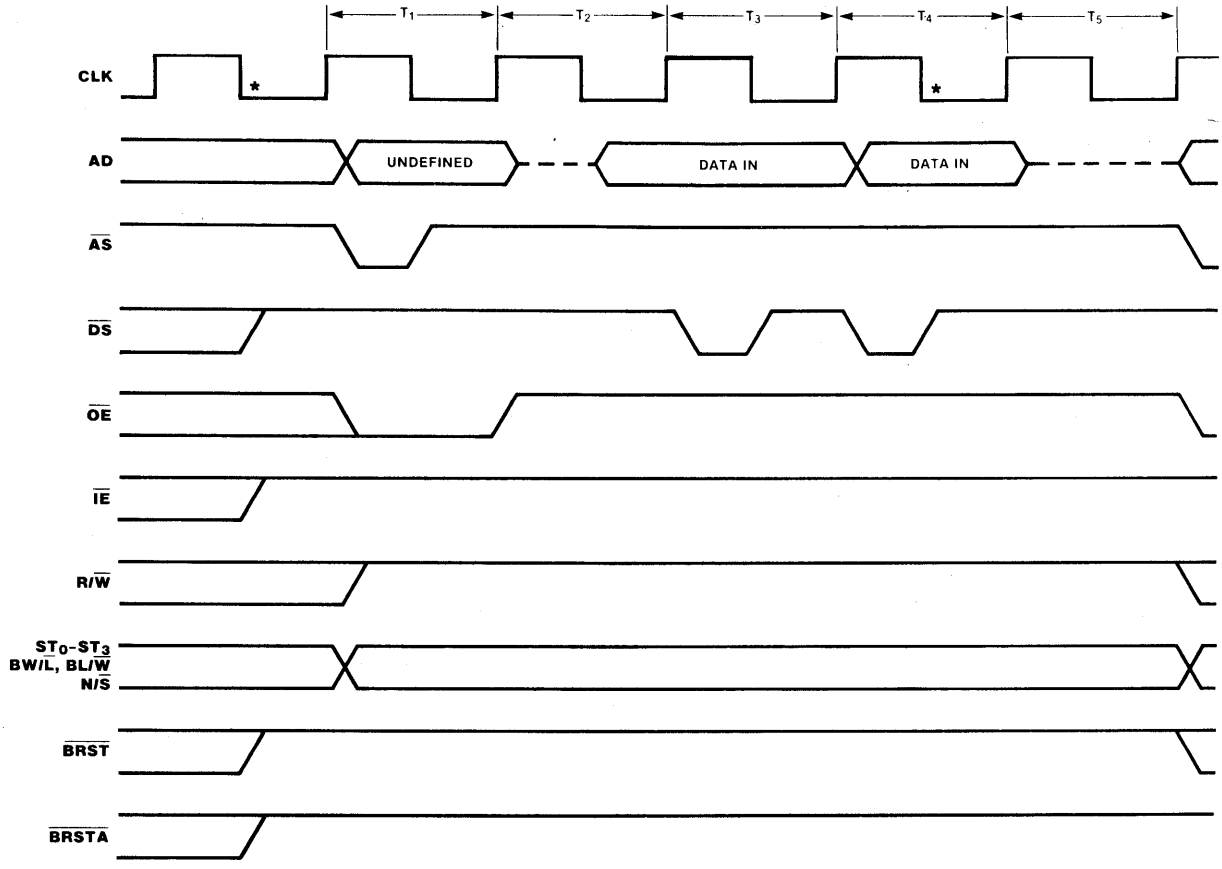
CPU-APU Data Transfer Transactions. Transactions to transfer data between the CPU and APU use status 0001. The number of words transferred is known to the CPU and EPU from the EPA instruction opcode (n-1 field). The transactions transfer one or more longwords of data until all words have been transferred. If the last transfer contains only a word, the data is replicated on AD_0–AD_{15} and AD_{16}–AD_{31}. The CPU does not assert $\overline{BRST}$ and ignores RSP_0–RSP_1 and $\overline{BRSTA}$.



*$\overline{EPUBSY}$ sampled.
+ $\overline{EPUBSY}$ sampled if EPU internal operation.

Figure 21. Z8070-Z80,000 Instruction Transfer

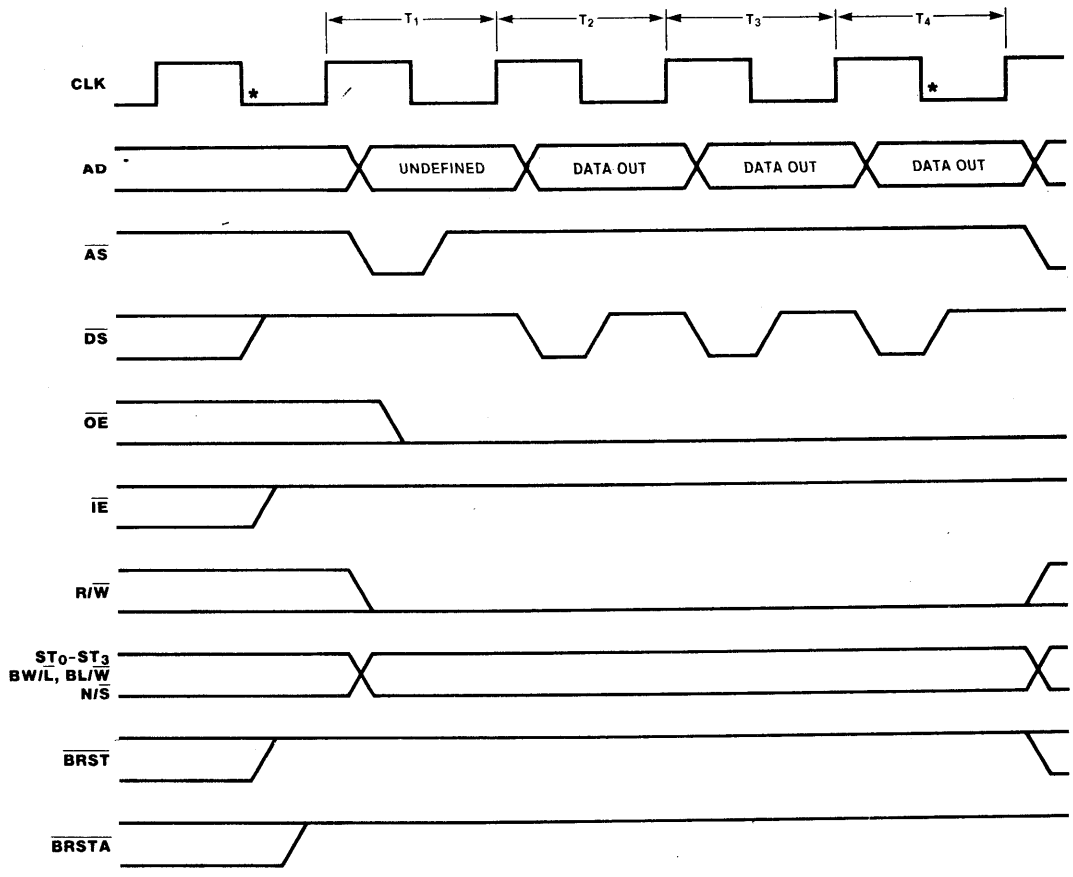
CPU-APU Data Read. Timing for a CPU-APU data read transaction is shown in Figure 22. This example has two data transfers; any number of data transfers between 1 and 4 is possible. The transaction begins with Address Strobe to indicate that the status and control signals are valid. The CPU stops driving the AD lines at the end of T_1, and the APU begins driving the AD lines in the middle of T_2. At the beginning of T_3 the CPU asserts $\overline{DS}$. In the middle of T_3 the CPU samples the data and negates $\overline{DS}$. The second longword of data is transferred during T_4. After the last data transfer the CPU inserts an idle bus cycle (T_5 in the example) during which neither the CPU nor APU drive the AD lines.



*$\overline{EPUBSY}$ sampled.

Figure 22. Z8070-Z80,000 Data Read Transaction

CPU-APU Data Write. Timing for a CPU-APU data write transaction is shown in Figure 23. This example has three data transfers; any number of data transfers between 1 and 4 is possible. Timing for the first transfer is identical to the CPU-APU instruction transfer transaction. A second longword of data is transferred during T_3, and the third longword is transferred during T_4.

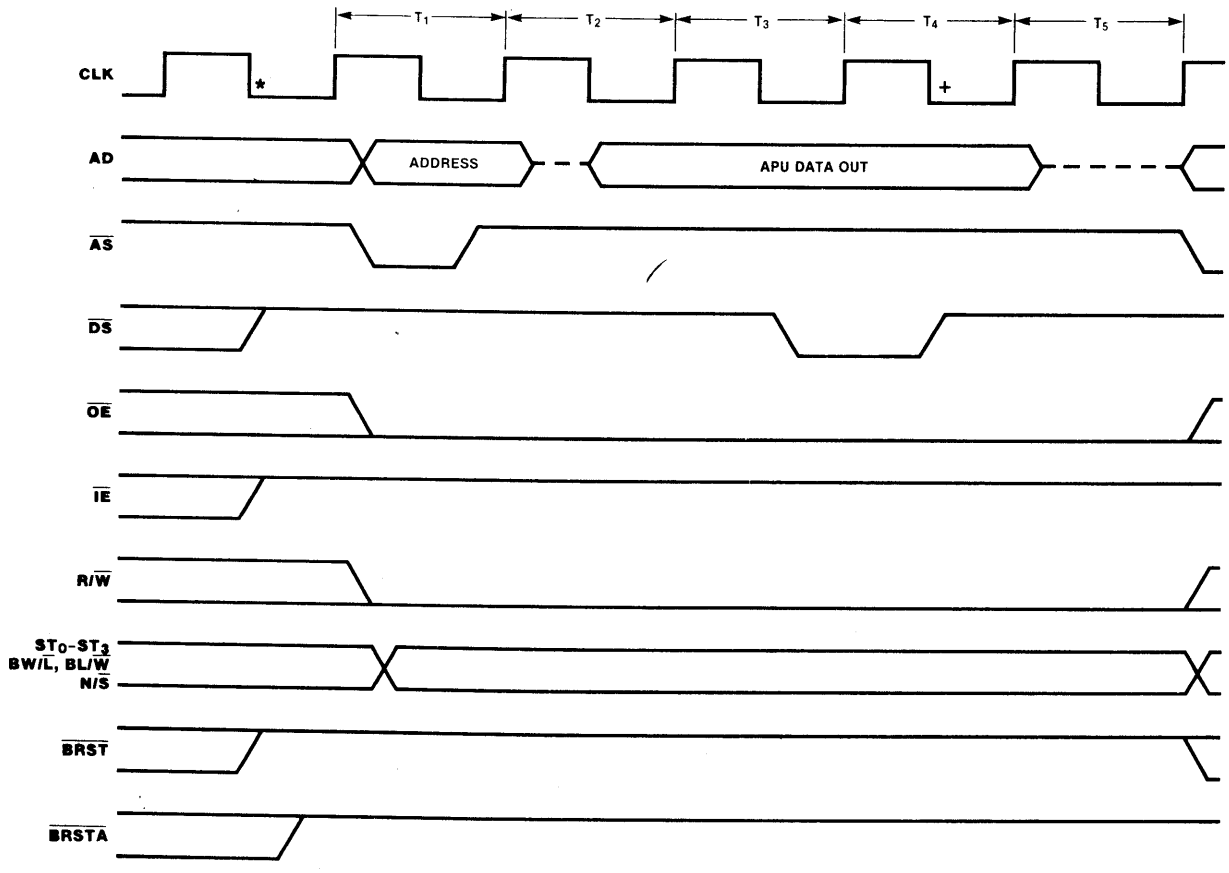


*EPUBSY sampled.

Figure 23. Z8070-Z80,000 Data Write Transaction

APU-Memory Transactions. The CPU uses status 1010 or 1011 for the APU to read from and write to memory. The timing for an APU-memory read is identical to a CPU-memory read. The APU passively monitors the CPU timing on the bus, and uses the two least-significant address bits, the data transfer size, and the length of the operand from the instruction to select the bytes it needs from the AD line.

The timing for an APU-memory write differs slightly from a CPU-memory write. Two extra bus cycles are included to pass the AD lines from CPU to APU after the address transfer and from APU back to CPU after the last data transfer. An example is shown (Figure 24) for a single APU-memory data transaction with no wait states. The CPU stops driving the AD lines at the end of T_1, and the APU begins driving the AD lines in the middle of T_2. $\overline{DS}$ is asserted in the middle of T_3, one bus cycle later than for CPU-memory write timing. The CPU negates $\overline{DS}$ in the middle of T_4. The CPU can insert wait states in the middle of T_4. The APU continues to drive the AD lines until the end of T_4. After the last data transfer the CPU inserts an idle bus cycle (T_5 in the example) during which neither the CPU nor APU drive the AD lines. APU-memory burst write transactions are similarly extended by two bus cycles compared with CPU-memory burst write timing. One cycle is inserted before the first data transfer, and another cycle after the last data transfer.



*$\overline{EPUBSY}$ sampled.
+ RSP_0-RSP_1 sampled; $\overline{EPUBSY}$ sampled if last transaction.

Figure 24. Z8070-Memory Single Write Timing (Z80,000)

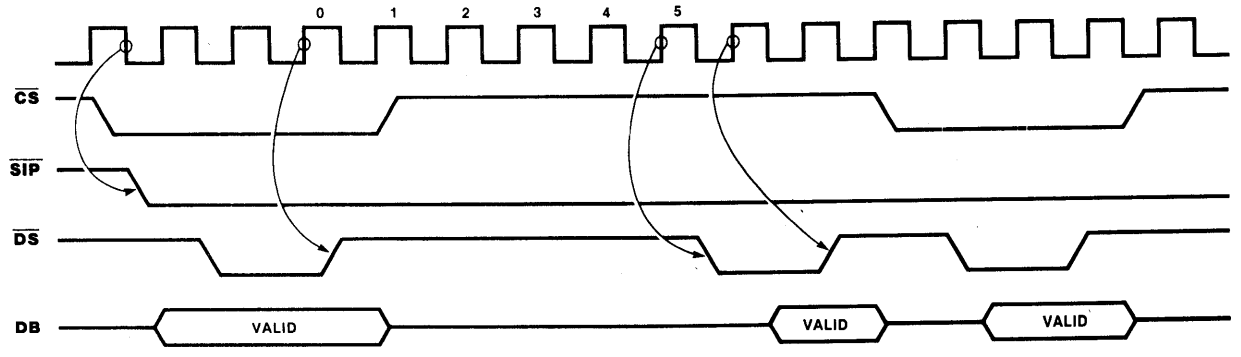
Universal Bus Transactions. The Universal interface is designed to accommodate a wide variety of CPUs. It can be tailored for data bus sizes of 8 bits, 16 bits and 32 bits. In transfers of data strings between the Z8070 and memory or the CPU, the MSB is presented first, and the data is left-justified, starting on pin DB_0, when odd numbers of strings are being presented. For example, a Z8070/memory transaction involving the double extended number 123456789ABCDEF02468 would be sent on a long-word (32 bit) bus as follows:

first longword 12345678
second longword 9ABCDEF0
third longword 2468xxxx
(where x = don't care)

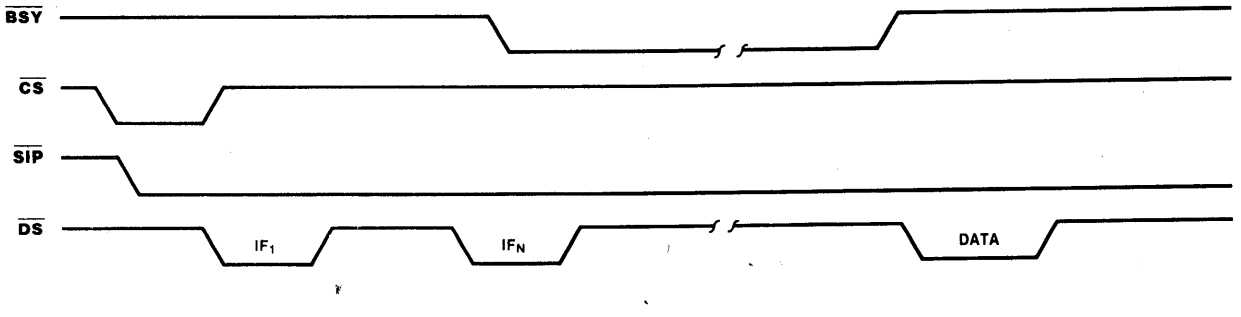
The CPU samples $\overline{SIP}$ and $\overline{BSY}$ before initiating an instruction transfer to the APU. If both signals are inactive, the CPU asserts $\overline{CS}$ and controls the bus to transfer the floating-point instruction and any associated data. The APU recognizes that an instruction is for it when both $\overline{CS}$ and $\overline{DS}$ are asserted. The APU then asserts $\overline{SIP}$ to signal that an APU sequence is in progress. $\overline{SIP}$ stays Low throughout the bus transaction portion of instruction processing. This prohibits the CPU from initiating another floating-point instruction transfer until the APU's Interface Processor is empty. If an instruction is executing in the Data Processor and another instruction is queued in the Interface Processor, the APU asserts $\overline{BSY}$ until the Interface Processor is empty. As with other CPU options, the APU asserts $\overline{BSY}$ on receiving any instruction involving a store. Figure 25 illustrates Universal Interface bus transactions.

Exception Processing

When the Z8070 detects an exception (and the particular interrupt mask and master interrupt enable are enabled) it asserts $\overline{INT}$. If there is an instruction in the instruction queue that was queued or being queued when the interrupt occurred, this queued instruction will be invalidated ("flushed" without being executed). If the Z8070 has $\overline{BSY}$ asserted, (e.g., the queued instruction is a store) then $\overline{BSY}$ will be negated so that the store instruction can have the Z8070-memory transaction completed. This stores undefined data in memory, and the store instruction is effectively invalidated. If an instruction has been invalidated, the RPC bit in the System Configuration register is set, indicating that the instruction was invalidated and must be re-executed. If an interrupt occurs and the queue is empty, RPC is not modified.



Typical Z8070 Universal Interface Transaction



BSY Function During Universal Interface Store

Figure 25. Universal Interface Transfer Transaction

Z80,000™ CPU

Zilog

NEW 1984

Preliminary Product Specification

September 1983

FEATURES

- Full 32-bit architecture and implementation
- 4 gigabytes (billion bytes) of directly addressable memory
- Linear or segmented address space
- Virtual memory management integrated with CPU
- On-chip cache memory
- General-purpose register file with 16 32-bit registers
- 9 general addressing modes
- Numerous data types include bit, bit field, logical value, signed integer, and string
- Mainframe performance
- Extended Processing Architecture supports floating point operations
- Regular use of operations, addressing modes, and data types in instruction set
- System and Normal modes of operation with separate stacks
- Sophisticated interrupt and trap handling
- Software is a binary-compatible extension of Z8000™ software
- Hardware is compatible with other Z-BUS™ components

GENERAL DESCRIPTION

The Z80,000 CPU is an advanced, high-end 32-bit microprocessor that integrates the architecture of a mainframe computer into a single chip. While maintaining full compatibility with Z8000 family software and hardware, the Z80,000 CPU offers greater power and flexibility in both its architecture and interface capability. Operating systems and compilers are easily developed in the Z80,000 CPU's high-quality environment, and the hardware interface provides for connection to a wide variety of system configurations.

Addresses in the Z80,000 CPU are 32 bits. This allows direct addressing of 4G bytes in each of four address spaces: System-mode data, System-mode instruction, Normal-mode data, and Normal-mode instruction. The CPU supports three modes of address representation. The 16-bit compact addresses are compatible with Z8000 nonsegmented mode. The 32-bit segmented addresses include both 16-bit offset, which is compatible with Z8000 segmented mode, and 24-bit offset. In addition a full 32-bit linear address space is provided.

The CPU features a general-purpose register file with sixteen 32-bit registers, and nine operand addressing modes. The various addressing modes allow encoding choices for compact representation or for full 32-bit addressing. The instruction set can operate on bit, bit field, logical value, signed integer, unsigned integer, address, string, stack, and packed decimal byte data types. Logical and arithmetic instructions operate on bytes (8 bits), words (16 bits) and longwords (32 bits). The Extended Processing Architecture (EPA) supports floating point operations. In addition, the instruction set is highly regular in combining operations, data types, and addressing modes. High-level language compilation is supported with instructions for procedure linkage, array index calculation, and bounds checking. Other instructions provide operating system functions such as system call and control of memory management.

There are two main operating modes, System and Normal, supported by separate stacks. User programs operate in Normal mode, while sensitive operating

system functions are performed in System mode. This protects critical parts of the operating system from user access. In addition, some instructions are privileged, and execute only in System mode. Memory management functions protect both system memory from user programs, and user memory from other users. Vectored, nonvectored, and nonmaskable interrupts support real-time operating systems.

Memory management is fully integrated with the CPU; no external support circuitry is necessary. A paging address translation mechanism is implemented. Registers in the CPU point to address translation tables located in memory; the most recently used table entries are kept in a Translation Lookaside Buffer in the CPU. The CPU performs logical to physical address translation and access protection for each memory reference. When a logical memory reference causes a translation or protection violation, the state of the CPU is automatically restored to restart the instruction. I/O ports can be referenced either by dedicated instructions or by the memory management mechanism mapping logical memory addresses to I/O port addresses.

Extensive trapping facilities, such as integer overflow, subrange out of bounds, and subscript out of bounds, catch common run-time errors. Software debuggers can use trace and breakpoint traps. Privileged instruction traps and memory protection violation traps secure the operating system from user programming errors or mischief. The Overflow Stack allows recovery from otherwise fatal errors.

The CPU has full 32-bit internal address and data paths. Externally, 32 pins time-multiplex the address and data. The interface is compatible with the complete line of Z-BUS peripherals. The hardware interface features 16-bit or 32-bit memory data path and programmable Wait states. Burst transfers and an on-chip cache for instructions and data help develop high-performance systems. The interface supports multiprocessing configurations with interlocked memory references and two types of bus request protocols. The system designer can tailor the Z80,000-based system to cost and performance needs.

In summary, the Z80,000 CPU meets and surpasses the requirements of medium and high-end microprocessor systems for the 1980s. Software program development is easily accomplished with the CPU's sophisticated architecture. The highly pipelined design, on-chip cache, and external interface support systems ranging from dedicated controllers to mainframe computers. While Zilog continues to develop support for the Z80,000 CPU, Z8000 peripherals and development software are fully compatible with this latest in Zilog's line of high-performance microprocessors.

REGISTERS

The Z80,000 CPU is a register-oriented processor offering sixteen 32-bit general-purpose registers, a 32-bit Program Counter (PC), a 16-bit Flag and Control Word (FCW), and nine other special-purpose registers.

The general-purpose register file (Figure 1) contains 64 bytes of storage. The first 16 bytes (RL0,RH0,...,RL7,RH7) can be used as accumulators for byte data. The first 16 words (R0,R1,...,R15) can be used as accumulators for word data, as index registers (except R0), or for memory addresses in compact mode (except R0). Any longword register (RR0,RR2,...,RR30) can be used as an accumulator for longword data, an index register (except RR0), or for memory addresses in linear or segmented modes (except RR0). Quadword registers (RQ0,RQ4,...,RQ28) can be used as accumulators for Multiply, Divide, and Extend Sign instructions. This unique register organization allows bytes and words of data to be manipulated conveniently while leaving most of the register file free to hold addresses, counters, and any other data.

Two registers are dedicated to the Stack Pointer (SP) and Frame Pointer (FP) used by Call, Enter, Exit, and Return instructions. In compact mode, R15 is the Stack Pointer and R14 the Frame Pointer. In linear or segmented mode, RR14 is the Stack Pointer and RR12 is the Frame Pointer.

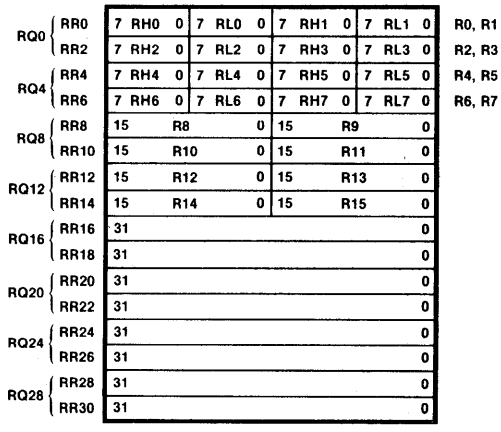


Figure 1. General-Purpose Register File

The PC and FCW form the Program Status (Figure 2), which is automatically saved for traps and interrupts. The bits in FCW indicate operating modes, masks for traps and interrupts, and flags set according to the result of instructions. The remaining special registers are used for memory management, system configuration, and other CPU control (Figure 3).

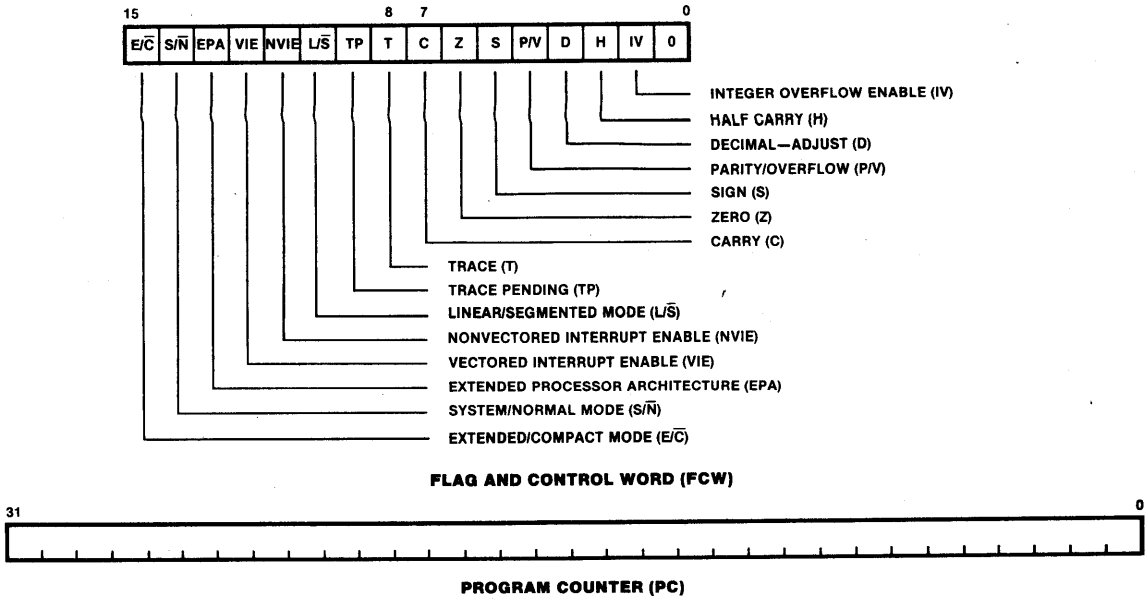


Figure 2. Program Status Registers

ADDRESS SPACES

As shown in Figure 4, the CPU has three modes of address representation: compact, segmented, and linear. The mode is selected by two control bits in the Flag and Control Word register (see Table 1). The Extended/Compact (E/C̄) bit selects whether compact addresses (16 bits) or extended addresses (32 bits) are used. For extended addresses the Linear/Segmented (L/S̄) bit selects whether linear or segmented addresses are used.

The Load Address instruction can be used to manipulate addresses in any mode of representation.

In compact mode, addresses are 16 bits. Address calculations using compact addresses involve all 16 bits. Compact mode is more efficient and less program-consuming for applications requiring less than 64K bytes of program and less than 64K bytes of data. This efficiency is due to shorter instructions in compact mode, and the fact that addresses in the register file use word rather than longword registers. Applications requiring more than 64K bytes of either program or data should use segmented or linear modes.

Table 1. Address Representation

Control Bits in FCW E/C̄	L/S̄	Representation
0	0	Compact
0	1	Reserved
1	0	Segmented
1	1	Linear

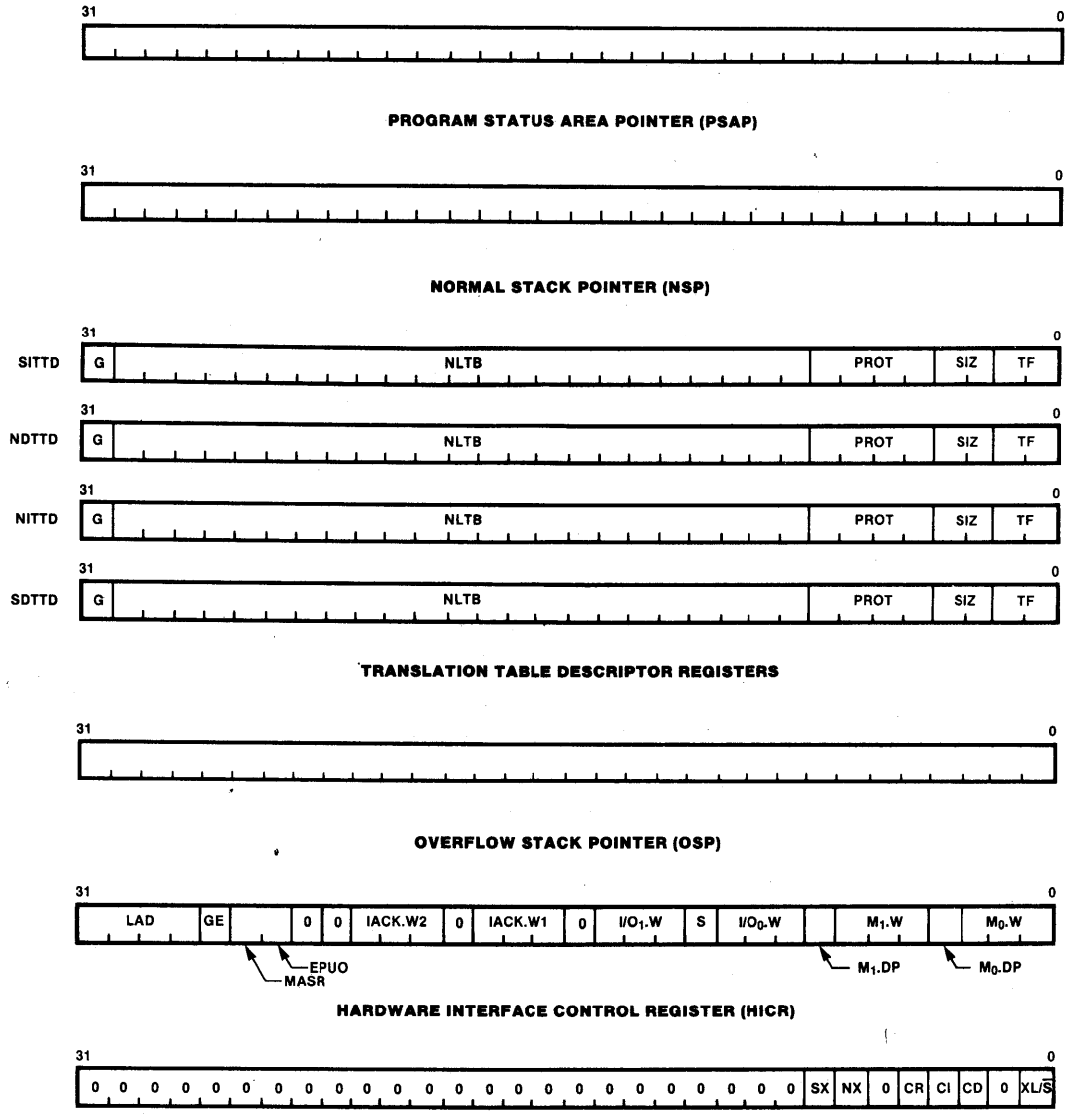


Figure 3. Special-Purpose Control Registers

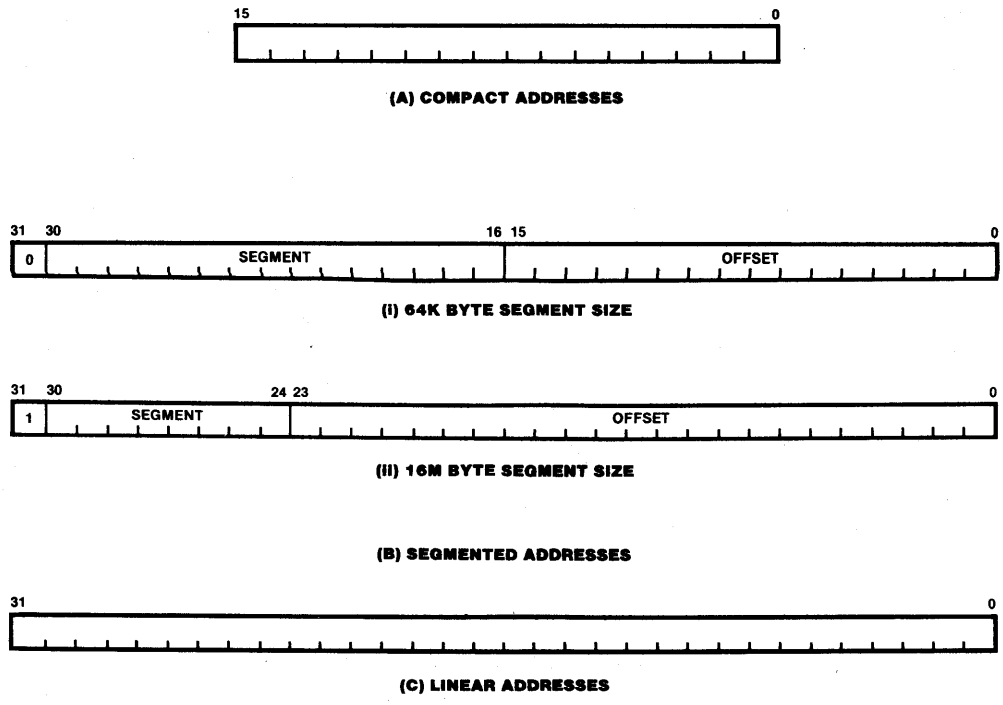


Figure 4. Address Representations

In segmented mode, addresses are 32 bits. Segmented addresses are composed of either a 15-bit segment number and a 16-bit segment offset or a 7-bit segment number and a 24-bit segment offset. Bit 31 of the address selects either of the two types of segmented addresses. Address calculations using segmented addresses involve only the segment offset; the segment number is unaffected. In segmented mode, the address space allows up to 32,768 segments of 64K byte maximum size and up to 128 segments of 16M byte maximum size. Many applications benefit from the logical structure of segmentation by allocating individual objects, such as a program module, stack, or large data structure, to separate segments.

In linear mode, addresses are 32-bits. Address calculations using linear addresses involve all 32 bits. In linear mode, the address space of 4G bytes is uniform and unstructured. Many applications benefit from the flexibility of linear addressing by allocating objects at arbitrary positions in the address space.

Memory is byte addressable by the CPU. The address used for multiple-byte data is the address of the most significant byte. Multiple-byte data can be located at any byte address with no alignment restrictions.

I/O ports can be addressed by either dedicated instructions or by the memory management mechanism mapping logical memory addresses to I/O ports. I/O ports can be byte, word, or longword in size.

NORMAL AND SYSTEM MODES

The CPU has two modes of operation, Normal and System, selected by the S/$\overline{N}$ bit in the Flag and Control Word register. These modes impact on CPU operation in three areas: privileged instructions, stack pointers, and memory management.

Since the most sensitive portions of the operating system usually execute in System mode, separate stack pointers are used to isolate the two operating modes.

Some instructions, such as those performing I/O operations or accessing control registers, can only be executed in System mode; in addition, the memory management mechanism allows access to some memory locations in System mode only. Programs executing in Normal mode can request services from the operating system using the System Call instruction and trap.

MEMORY MANAGEMENT

The CPU and the operating system cooperate in translating logical to physical addresses and protecting memory for execute, read, and write accesses. The CPU implements a paging translation mechanism similar to that in most mainframe and super-minicomputers. The operating system creates translation tables in memory, then initializes pointers to the tables in control registers. The CPU automatically references the tables to perform address translation and access protection. The CPU enables the operating system to implement efficient virtual memory by marking pages that have been referenced or modified and by automatically recovering from address translation faults to allow instruction restart.

The paging translation scheme implemented by the CPU divides the logical address spaces into pages and the physical address space into frames. The logical pages and physical frames are each 1K bytes. A logical page, which is specified by the 22 most significant bits of the logical address, can be mapped into any physical frame, which is specified by the 22 most significant bits of the physical address. The 10 least significant bits, which specify the byte within a page or frame, are not translated. For each memory reference, the CPU translates the logical address to the corresponding physical address and also tests whether access to the memory location is permitted. For most references the information needed to perform the translation is stored in the CPU Translation Lookaside Buffer (TLB). The TLB (Figure 5) stores the translation information for the 16 most recently referenced pages in a fully associative memory. Only when information to translate the page is missing from the TLB does the CPU reference translation tables in memory. In the case of a TLB miss, the CPU translates the logical address using the procedure described below and the translation information is loaded into the TLB entry of the least recently referenced page.

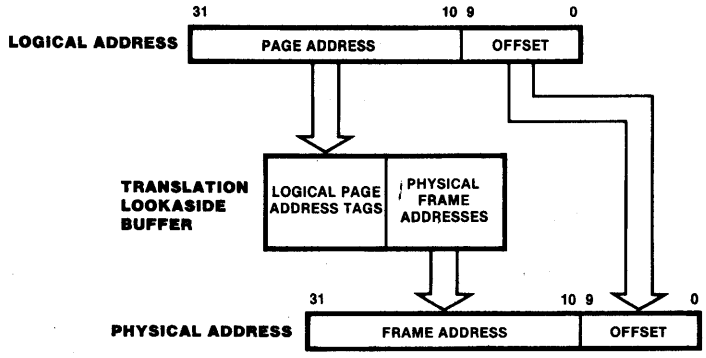


Figure 5. Address Translation Using the TLB

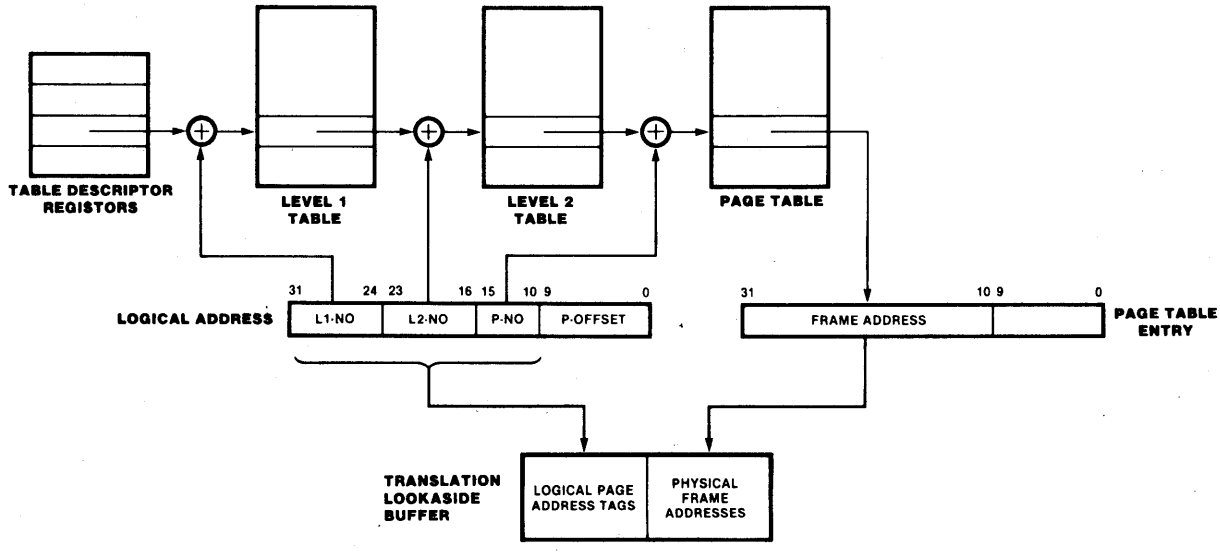


Figure 6. Automatic Loading of the TLB Using Tables in Memory

The address translation mechanism is a three-level paging scheme. A logical address is partitioned into an 8-bit level-1 field (L1-NO), an 8-bit level-2 field (L2-NO), a 6-bit page number field (P-NO), and a 10-bit page offset field (P-OFFSET). During translation, the L1-NO, L2-NO, and P-NO fields are used as indexes into tables in physical memory. The TF field of the Segment Table Descriptor registers can be programmed to selectively skip the first and second-level tables to reduce both the storage space needed for tables and the number of references necessary to perform translation when the information to translate a page is missing from the on-chip TLB.

To load the TLB (Figure 6), the CPU selects one of four table descriptor registers according to the address space for the reference: system instruction, system data, normal instruction, or normal data. The table descriptor register points to the beginning of the level-1 table in memory; the L1-NO field of the logical address is used as an index into this table to select the level-1 table entry. Next, the level-1 table entry points to the beginning of the level-2 table; the L2-NO field of the logical address is used as an index into this table to select the level-2 table entry. After this, the level-2 table entry points to the beginning of the page table in memory; the PAGE-NO field of the logical address is used as an index into this table to select the page table entry. The page table entry contains the physical address of the frame corresponding to the logical address. The CPU then loads the logical page address and physical frame address into the TLB.

When bit 31 in the page table entry is 1, the frame is in physical I/O space. The CPU uses I/O status and timing for the reference. Thus, the address translation process allows protected access to memory-mapped I/O devices.

Figures 7 and 8 show the translation and table entry formats.

Access protection information (Table 2) is encoded in the 4-bit PROT field contained in the Translation Table Descriptor, level-1 table entry, level-2 table entry, or page table entry. During the translation process, a PROT field is encountered at each level. The first PROT field with value other than 1000 is selected; the other PROT fields are ignored. The protection code specifies the types of access (execute, read, and write) permitted in Normal and System modes.

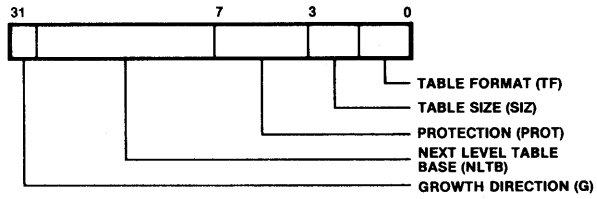


Table Format (TF)

00 THREE LEVELS
01 SKIP LEVEL 2 TABLES
10 SKIP LEVEL 1 TABLES
11 SKIP LEVEL 1 AND LEVEL 2 TABLES

TABLE SIZE (SIZ)	VALID TABLE ENTRIES G = 0	G = 1
00	0 TO 63	0 TO 255
01	0 TO 127	64 TO 255
10	0 TO 191	128 TO 255
11	0 TO 255	192 TO 255

Figure 7. Translation Table Descriptor

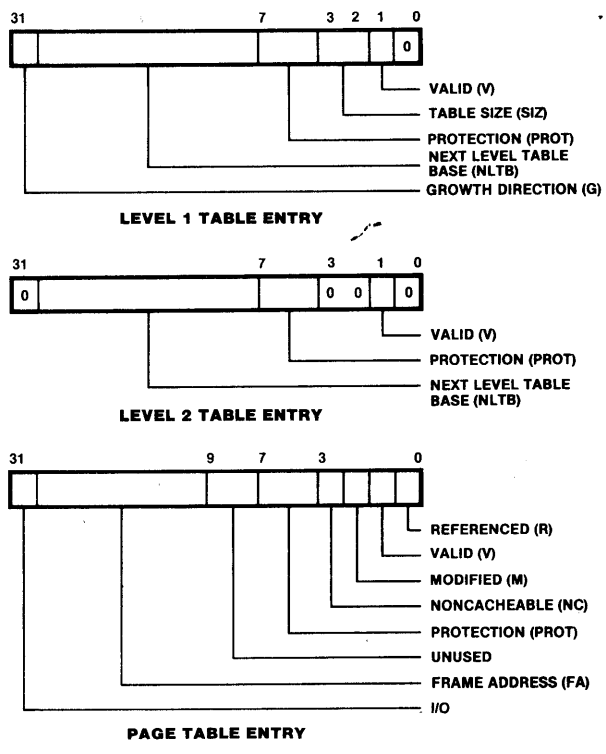


Figure 8. Table Entry Formats

Table 2. Protection Field Encoding

Encoding	System	Normal
0000	NA	NA
0001	RE	NA
0010	RE	E
0011	RE	RE
0100	E	NA
0101	E	E
0110	R	NA
0111	R	R
1000	Note	Note
1001	RW	NA
1010	RW	R
1011	RW	RW
1100	RWE	NA
1101	RWE	E
1110	RWE	RE
1111	RWE	RWE

NA—no access is permitted
R—read access is permitted
W—write access is permitted
E—execute access is permitted
Note—use the protection field of the next level translation table or NA (for page tables)

There are several optional features that allow the number of levels and the size of tables to be reduced. When memory address spaces are not separated, two or more of the translation table descriptor registers can be loaded with the same value so that tables are held in common. The table descriptor register can specify that either or both of the level-1 and level-2 tables should be skipped during the translation process. Level-1 tables can be skipped when a 24-bit logical address space is sufficient, both level-1 and level-2 segment tables can be skipped for compact addresses, and level-2 tables can be skipped for compatibility with Z8000 segmented addresses. The table size can be reduced by allocating only 256, 512, or 768 bytes for the tables; the remaining table entries are assumed invalid. The tables can be allocated efficiently for downward growing stacks by setting the G bit of the translation table descriptor or level-1 table entry.

During execution of an instruction, if an invalid translation table entry is encountered or a protection violation is detected, the CPU traps to the operating system. The CPU automatically saves the state of registers and memory so the instruction can simply be restarted.

Several instructions are provided to help the operating system control memory management. The Purge TLB instructions are used to purge the Translation Lookaside Buffer of a single entry, Normal mode entries, or all entries. The Load Physical Address instructions translate logical addresses into physical addresses, and set the flags to verify access permission for system call parameters. The instructions, Load Normal Data and Load Normal Instruction allow System mode programs to reference Normal memory spaces.

The memory management mechanism can be selectively enabled for normal and system space references by using the SX and NX bits of the System Configuration Control Longword register.

EXCEPTIONS

The CPU supports four types of exceptions: Reset, Bus Error, interrupts, and traps. A reset exception occurs when the $\overline{\text{RESET}}$ line is activated; this causes the CPU to be reset to an initialized state. A bus error exception occurs when external hardware indicates an error on a bus transaction. An interrupt is an asynchronous event that typically occurs when a peripheral device needs attention. A trap is a synchronous event that occurs when a particular condition is detected during execution of an instruction.

In responding to a reset exception, the CPU fetches the program status (FCW and PC) from physical address 2. In responding to other exceptions, the CPU pushes the old program status onto the system stack along with information specific to the type of exception. The CPU then fetches a new program status from the table designated by the Program Status Area Pointer control register.

During exception processing, the mode of address representation for the system Stack Pointer and Program Status Area Pointer is either linear or segmented, selected by the XL/S bit in the System Configuration Control Longword register. Three types of interrupts are supported: vectored, nonvectored, and nonmaskable. The vectored and nonvectored interrupts have mask bits in the FCW. All interrupts read an identifier word from the bus during an Interrupt Acknowledge transaction and save the word on the system stack. Vectored interrupts use the lower byte of this word to select a unique PC value from the Program Status Area.

The CPU recognizes twelve trap conditions.

- Extended Instruction Trap occurs when an Extended Processing Architecture instruction is executed and the EPA bit in the FCW is clear.
- Privileged Instruction Trap occurs when an attempt is made to execute a privileged instruction in Normal mode.
- System Call Trap occurs when the System Call instruction is executed to request service from the operating system.
- Address Translation Trap occurs when an address translation or access protection violation is detected.
- Reserved Instruction Trap occurs when an attempt is made to execute an instruction with a reserved bit pattern.
- Odd PC Trap occurs when an odd-byte address is loaded into the PC.
- Trace Trap occurs after execution of an instruction when tracing is enabled by setting the T bit in the FCW.
- Breakpoint Trap occurs when the Breakpoint instruction is executed, usually to invoke a debugging or monitoring program.
- Conditional Trap occurs when the Conditional Trap instruction is executed and the specified condition code is satisfied. This trap can allow detection of user-defined exceptions.
- Integer Overflow Trap occurs when overflow is detected during execution of an integer arithmetic instruction and the IV bit in the FCW is set.
- Bounds Check Trap occurs when the Check instruction is executed and the source operand is out of bounds.
- Subscript Error Trap occurs when the Index instruction is executed and the subscript operand is out of bounds.

In descending order, the priority of exceptions is: reset, bus error, trap (other than trace), nonmaskable interrupt, vectored interrupt, and nonvectored interrupt. Trace Trap uses two control bits, T and TP, so that when tracing is enabled, exactly one trace trap occurs after each instruction is executed.

When an address translation trap occurs for the system stack, the CPU cannot save the program status and other exception information on the system stack. The system can still recover from this otherwise fatal error because the CPU saves the information on the Overflow Stack designated in physical memory by the Overflow Stack Pointer control register.

ADDRESSING MODES

The CPU locates operands (the data manipulated by instructions) in registers, memory, I/O ports, or in the instruction. The location of an operand is specified by one of nine addressing modes (Figure 9): Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Index (X), Base Address (BA), Base Index (BX), Relative Address (RA), and Relative Index (RX). Most operations can be used with any addressing mode; however, some operations are restricted. Instruction encoding provides compact representation for the most frequently used addressing modes.

The term Extended Addressing Modes (EAM) refers to the following addressing modes that require one or more extra words to be added to the opcode.

- In compact mode: DA and X (X is equivalent to BA)
- In linear or segmented modes: DA, X, BA, BX, RA, and RX

Addressing Mode	Operand Addressing			Operand Value
	In the Instruction	In a Register	In Memory	
R Register	REGISTER ADDRESS → OPERAND			The content of the register
IM Immediate	OPERAND			In the instruction
***IR** Indirect Register	REGISTER ADDRESS → ADDRESS → OPERAND			The content of the location whose address is in the register
DA Direct Address	ADDRESS → OPERAND			The content of the location whose address is in the instruction
***X** Index	REGISTER ADDRESS, BASE ADDRESS; INDEX; (+) → OPERAND			The content of the location whose address is the address in the instruction, plus the content of the Index register
RA Relative Address	DISPLACEMENT; PC VALUE; (±) → OPERAND			The content of the location whose address is the address in the instruction, plus the content of the Index register
***BA** Base Address	REGISTER ADDRESS, DISPLACEMENT; BASE ADDRESS; (+) → OPERAND			The content of the location whose address is the content of the Base register, plus the displacement in the instruction
***BX** Base Index	REGISTER ADDRESS, REGISTER ADDRESS, DISPLACEMENT; BASE ADDRESS, INDEX; (+) → OPERAND			The content of the location whose address is the content of the Base register, plus the content of the Index register, plus the displacement in the instruction
***RX** Relative Index	REGISTER ADDRESS, DISPLACEMENT; PC VALUE, INDEX; (+) → OPERAND			The content of the location whose address is the content of the Program Counter, plus the content of the Index register, plus the displacement in the instruction.

*R0 and RR0 cannot be used for Indirect, Base, or Index registers

Figure 9. Addressing Modes

DATA TYPES

The CPU supports operations on the following data types.

- Bit
- Bit field—1 to 32 contiguous bits within a longword
- Signed integer—byte, word, longword, and quadword
- Unsigned integer—byte, word, longword, and quadword
- Logical value—byte, word, and longword
- Address
- Packed BCD integer—byte
- Stack—word and longword
- String—dynamic length byte, word, and longword

FLAGS AND CONDITION CODES

Arithmetic, logical, and many other instructions affect the six flag bits (C, Z, S, P/V, D, and H) in the FCW to provide information about an operation's result. Generally, C indicates carry or borrow from the result, Z indicates the result is zero, S indicates whether the result is negative or positive, and P/V indicates parity or overflow. D and H are used for decimal arithmetic.

Jump, Test Condition Code, and several other instructions test the state of the flags. The conditions that can be tested are shown in Table 3.

Table 3. Flags and Condition Codes

Code	Meaning	Flag Setting	Binary
F	Always false	—	0000
T	Always true	—	1000
Z	Zero	Z = 1	0110
NZ	Not zero	Z = 0	1110
C	Carry	C = 1	0111
NC	No carry	C = 0	1111
PL	Plus	S = 0	1101
MI	Minus	S = 1	0101
NE	Not equal	Z = 0	1110
EQ	Equal	Z = 1	0110
OV	Overflow	V = 1	0100
NOV	No overflow	V = 0	1100
PE	Parity even	P = 1	0100
PO	Parity odd	P = 0	1100
GE	Greater than or equal	(S XOR V) = 0	1001
LT	Less than	(S XOR V) = 1	0001
GT	Greater than	(Z OR (S XOR V)) = 0	1010
LE	Less than or equal	(Z OR (S XOR V)) = 1	0010
UGE	Unsigned greater than or equal	C = 0	1111
ULT	Unsigned less than	C = 1	0111
UGT	Unsigned greater than	((C = 0) AND (Z = 0) = 1	1011
ULE	Unsigned less than or equal	(C OR Z) = 1	0011

Some of the condition codes correspond to identical flag settings: i.e., Z-EQ, NZ-NE, NC-UGE, PE-OV, PO-NOV.

INSTRUCTION SET SUMMARY

The CPU provides 11 types of instructions:

- Load and Exchange
- Arithmetic
- Logical
- Program Control
- Bit Manipulation
- Rotate and Shift
- Block Transfer and String Manipulation
- Input/Output
- CPU Control
- Bit Field
- Extended Instructions

Instructions are encoded in one or more words, located in memory at even addresses.

Load and Exchange

Mnemonic	Operands	Addressing Modes	Operation
CLR **CLRB** **CLRL**	dst	dst: R,IR,EAM	**Clear** dst ← 0
CVTBW **CVTBL** **CVTWB** **CVTWL** **CVTLB** **CVTLW**	dst,src	dst: R src: R,IR,EAM or dst: IR,EAM src: R	**Convert** dst ← convert (src)
CVTUBW **CVTUBL** **CVTUWB** **CVTUWL** **CVTULB** **CVTULW**	dst,src	dst: R src: R,IR,EAM or dst: IR,EAM src: R	**Convert Unsigned** dst ← convert (src)
EX **EXB** **EXL**	dst,src	dst: R src: R,IR,EAM	**Exchange** dst ↔ src
LD **LDB** **LDL**	dst,src	dst: R src: R,IM,IR,EAM* or dst: IR,EAM* src: R,IM	**Load** dst ← src
LDA	dst,src	dst: R src: EAM*	**Load Address** dst ← Address (src)
LDAR	dst,src	dst: R src: RA	**Load Address Relative** dst ← Address (src)
LDK **LDKL**	dst,n	dst: R n: IM	**Load Constant** dst ← n (n = 0..15)

*Compact mode allows BX with no displacement for EAM.

Load and Exchange (Continued)

Mnemonic	Operands	Addressing Modes	Operation
LDM	dst,src,n	dst: R src: IM,IR,EAM n: IM or dst: IR,EAM src: R n: IM	**Load Multiple** dst ← src (n words)
LDML	mask,src or dst,mask	mask: IM src: IM,IR,EAM or dst: IR,EAM src: IM	**Load Multiple Long** dst (register mask) ← src or dst ← src (register mask)
LDR **LDRB** **LDRL**	dst,src	dst: R src: RA or dst: RA src: R	**Load Relative** dst ← src
POP **POPL**	dst,src	dst: R,IR,EAM src: IR	**Pop** dst ← src Autoincrement src address
PUSH **PUSHL**	dst,src	dst: IR src: R,IM,IR,EAM	**Push** Autodecrement dst address dst ← src

Arithmetic

Mnemonic	Operands	Addressing Modes	Operation
ADC **ADCB** **ADCL**	dst,src	dst: R src: R	**Add with Carry** dst ← dst + src + C
ADD **ADDB** **ADDL**	dst,src	dst: R src: R,IM,IR,EAM	**Add** dst ← dst + src
CHK **CHKB** **CHKL**	dst,src	dst: R src: IM,IR,EAM	**Check** compare dst with src bounds if out of bounds then trap
CP **CPB** **CPL**	dst,src	dst: R src: R,IM,IR,EAM or dst: IR,EAM src: IM	**Compare** dst – src
DAB	dst	dst: R	**Decimal Adjust**
DEC **DECB** **DECL**	dst,n	dst: R,IR,EAM n: IM	**Decrement** dst ← dst – n (n = 1..16)

Arithmetic (Continued)

Mnemonic	Operands	Addressing Modes	Operation
DECI **DECIB**	dst,n	dst: IR,EAM n: IM	**Decrement Interlocked** dst ← dst − n (n = 1..16)
DIV **DIVL**	dst,src	dst: R src: R,IM,IR,EAM	**Divide** dst (low) ← dst DIV src dst (high) ← dst REM src
DIVU **DIVUL**	dst,src	dst: R src: R,IM,IR,EAM	**Divide Unsigned** dst (low) ← dst DIV src dst (high) ← dst REM src
EXTS **EXTSB** **EXTSL**	dst	dst: R	**Extend Sign** dst ← sign_extend (dst (low))
INC **INCB** **INCL**	dst,n	dst: R,IR,EAM n: IM	**Increment** dst ← dst + n (n = 1..16)
INCI **INCIB**	dst,n	dst: IR,EAM n: IM	**Increment Interlocked** dst ← dst + n (n = 1..16)
INDEX **INDEXL**	dst,sub,src	dst: R sub: R src: IM,IR,EAM	**Index** calculate array index: check, scale, and accumulate
MULT **MULTL**	dst,src	dst: R src: R,IM,IR,EAM	**Multiply** dst ← dst (low) * src
MULTU **MULTUL**	dst,src	dst: R src: R,IM,IR,EAM	**Multiply Unsigned** dst ← dst (low) * src
NEG **NEGB** **NEGL**	dst	dst: R,IR,EAM	**Negate** dst ← dst
SBC **SBCB** **SBCL**	dst,src	dst: R src: R	**Subtract with Carry** dst ← dst − src − C
SUB **SUBB** **SUBL**	dst,src	dst: R src: R,IM,IR,EAM	**Subtract** dst ← dst − src
TESTA **TESTAB** **TESTAL**	dst	dst: R,IR,EAM	**Test Arithmetic** dst ← 0

Logical

Mnemonic	Operands	Addressing Modes	Operation
AND **ANDB** **ANDL**	dst,src	dst: R src: R,IM,IR,EAM	**And** dst ← dst AND src
COM **COMB** **COML**	dst	dst: R,IR,EAM	**Complement** dst ← NOT dst
OR **ORB** **ORL**	dst,src	dst: R src: R,IM,IR,EAM	**Or** dst ← dst OR src
TCC **TCCB** **TCCL**	cc,dst	dst: R	**Test Condition Code** if cc then dst ← dst OR 1
TEST **TESTB** **TESTL**	dst	dst: R,IR,EAM	**Test** dst OR 0
XOR **XORB** **XORL**	dst,src	dst: R src: R,IM,IR,EAM	**Xor** dst ← dst XOR src

Program Control

Mnemonic	Operands	Addressing Modes	Operation
BRKPT			**Breakpoint** (breakpoint trap) Push PS onto System Stack Push instruction PS ← Breakpoint PS
CALL	dst	dst: IR,EAM	**Call** Autodecrement SP @SP ← PC PC ← Address (dst)
CALR	dst	dst: RA	**Call Relative** Autodecrement SP @SP ← PC PC ← Address (dst)
DJNZ **DBJNZ** **DLJNZ**	cnt,dst	cnt: R dst: RA	**Decrement and Jump if Not Zero** cnt ← cnt − 1 if cnt ≠ 0 then PC ← Address (dst)

Program Control (Continued)

Mnemonic	Operands	Addressing Modes	Operation
ENTER	mask,siz	mask: IM siz: IM	**Enter Procedure** Push registers (mask) Push FP Push mask Push 0 (exception handler) FP ← SP + siz update integer overflow mask
EXIT			**Exit Procedure** SP ← FP Pop exception handler Pop mask Pop FP Pop registers (mask) restore integer overflow mask
JP	cc,dst	dst: IR,EAM	**Jump** if cc then PC ← Address (dst)
JR	cc,dst	dst: RA	**Jump Relative** if cc then PC ← Address (dst)
RET	cc		**Return** if cc then PC ← @SP Autoincrement SP
SC	src	src: IM	**System Call** (system call trap) Push PS onto System Stack Push instruction PS ← System Call PS
TRAP	cc,src	src: IM	**Trap Conditional** if cc then (condition trap) Push PS onto System Stack Push instruction PS ← Conditional Trap PS

Bit Manipulation

Mnemonic	Operands	Addressing Modes	Operation
BIT **BITB** **BITL**	dst,src	dst: R,IR,EAM src: IM or dst: R src: R	**Test Bit** Z ← NOT dst (src)
RES **RESB** **RESL**	dst,src	dst: R,IR,EAM src: IM or dst: R src: R	**Reset Bit** dst (src) ← 0
SET **SETB** **SETL**	dst,src	dst: R,IR,EAM src: IM or dst: R src: R	**Set Bit** dst (src) ← 1
TSET **TSETB** **TSETL**	dst	dst: R,IR,EAM	**Test and Set** s ← dst (MSB) dst ← -1

Rotate and Shift

Mnemonic	Operands	Addressing Modes	Operation
RL **RLB** **RLL**	dst,n	dst: R n: IM	**Rotate Left** dst ← dst rotate left n bits (n = 1 or 2)
RLC **RLCB** **RLCL**	dst,n	dst: R n: IM	**Rotate Left through Carry** dst,C ← dst,C rotate left n bits (n = 1 or 2)
RLDB	link,dst	link: R dst: R	**Rotate Left Digit** dst,link (0:3) ← dst,link (0:3) rotate left 1 digit
RR **RRB** **RRL**	dst,n	dst: R n: IM	**Rotate Right** dst ← dst rotate right n bits (n = 1 or 2)
RRC **RRCB** **RRCL**	dst,n	dst: R n: IM	**Rotate Right through Carry** dst,C ← dst,C rotate right n bits (n = 1 or 2)

Rotate and Shift (Continued)

Mnemonic	Operands	Addressing Modes	Operation
RRDB	link,dst	link: R dst: R	**Rotate Right Digit** dst,link (0:3) ← dst,link (0:3) rotate right 1 digit
SDA **SDAB** **SDAL**	dst,src	dst: R src: R	**Shift Dynamic Arithmetic** dst ← dst arithmetic shift src bits
SDL **SDLB** **SDLL**	dst,src	dst: R src: R	**Shift Dynamic Logical** dst ← dst logical shift src bits
SLA **SLAB** **SLAL**	dst,n	dst: R n: IM	**Shift Left Arithmetic** dst ← dst arithmetic shift left n bits
SLL **SLLB** **SLLL**	dst,n	dst: R n: IM	**Shift Left Logical** dst ← dst logical shift left n bits
SRA **SRAB** **SRAL**	dst,n	dst: R n: IM	**Shift Right Arithmetic** dst ← dst arithmetic shift right n bits
SRL **SRLB** **SRLL**	dst,n	dst: R n: IM	**Shift Right Logical** dst ← dst logical shift right n bits

Block Transfer and String Manipulation

Mnemonic	Operands	Addressing Modes	Operation
CPD **CPDB** **CPDL**	dst,src, cnt,cc	dst: R src: IR cnt: R	**Compare and Decrement** dst – src Autodecrement src address cnt ← cnt – 1
CPDR **CPDRB** **CPDRL**	dst,src, cnt,cc	dst: R src: IR cnt: R	**Compare, Decrement, and Repeat** Repeat dst – scr Autodecrement src address cnt ← cnt – 1 Until cc is true or cnt = 0
CPI **CPIB** **CPIL**	dst,src, cnt,cc	dst: R src: IR cnt: R	**Compare and Increment** dst – src Autoincrement src address cnt ← cnt – 1

Block Transfer and String Manipulation (Continued)

Mnemonic	Operands	Addressing Modes	Operation
CPIR **CPIRB** **CPIRL**	dst,src, cnt,cc	dst: R src: IR cnt: R	**Compare, Increment, and Repeat** Repeat dst – src Autoincrement src address cnt ← cnt – 1 Unit cc is true or cnt = 0
CPSD **CPSDB** **CPSDL**	dst,src, cnt,cc	dst: IR src: IR cnt: R	**Compare String and Decrement** dst – src Autodecrement dst and src addresses cnt ← cnt – 1
CPSDR **CPSDRB** **CPSDRL**	dst,src, cnt,cc	dst: IR src: IR cnt: R	**Compare String, Decrement, Repeat** Repeat dst – src Autodecrement dst and src addresses cnt ← cnt – 1 Until cc is true or cnt = 0
CPSI **CPSIB** **CPSIL**	dst,src, cnt,cc	dst: IR src: SR cnt: R	**Compare String and Increment** dst – src Autoincrement dst and src addresses cnt ← cnt – 1
CPSIR **CPSIRS** **CPSIRL**	dst,src, cnt,cc	dst: IR src: IR cnt: R	**Compare String, Increment, Repeat** Repeat dst – src Autoincrement dst and src addresses cnt ← cnt – 1 Until cc is true or cnt = 0
LDD **LDDB** **LDDL**	dst,src,cnt	dst: IR src: IR cnt: R	**Load and Decrement** dst ← src Autodecrement dst and src addresses cnt ← cnt – 1
LDDR **LDDRB** **LDDRL**	dst,src,cnt	dst: IR src: IR cnt: R	**Load, Decrement, and Repeat** Repeat dst ← src Autodecrement dst and src addresses cnt ← cnt – 1 Unitl cnt = 0
LDI **LDIB** **LDIL**	dst,src,cnt	dst: IR src: IR cnt: R	**Load and Increment** dst ← src Autoincrement dst and src addresses cnt ← cnt – 1

Block Transfer and String Manipulation (Continued)

Mnemonic	Operands	Addressing Modes	Operation
LDIR **LDIRB** **LDIRL**	dst,src,cnt	dst: IR src: IR cnt: R	**Load, Increment, and Repeat** Repeat dst ← src Autoincrement dst and src addresses cnt ← cnt − 1 Until cnt = 0
TRDB	dst,src,cnt	dst: IR src: IR cnt: R	**Translate and Decrement** dst ← src [dst] Autodecrement dst address cnt ← cnt − 1
TRDRB	dst,src,cnt	dst: IR src: IR cnt: R	**Translate, Decrement, and Repeat** Repeat dst ← src [dst] Autodecrement dst address cnt ← cnt − 1 Until cnt = 0
TRIB	dst,src,cnt	dst: IR src: IR cnt: R	**Translate and Increment** dst ← src [dst] Autoincrement dst address cnt ← cnt − 1
TRIRB	dst,src,cnt	dst: IR src: IR cnt: R	**Translate, Increment, and Repeat** Repeat dst ← src [dst] Autoincrement dst address cnt ← cnt − 1 Until cnt = 0
TRTDB	src1,src2, cnt	src1: IR src2: IR cnt: R	**Translate, Test, and Decrement** RH1 ← src2 [src1] Autodecrement src1 address cnt ← cnt − 1
TRTDRB	src1,src2, cnt	src1: IR src2: IR cnt: R	**Translate, Test, Decrement, Repeat** Repeat RH1 ← src2 [src1] Autodecrement src1 address cnt ← cnt − 1 Until RH1 ≠ 0 or cnt = 0
TRTIB	src1,src2, cnt	src1: IR src2: IR cnt: R	**Translate, Test, and Increment** RH1 ← src2 [src1] Autoincrement src1 address cnt ← cnt − 1

Block Transfer and String Manipulation (Continued)

Mnemonic	Operands	Addressing Modes	Operation
TRTIRB	src1,src2, cnt	src1: IR src2: IR cnt: R	**Translate, Test, Increment, Repeat** Repeat RH1 ← src2 [src1] Autoincrement src1 address cnt ← cnt − 1 Until RH1 ≠ 0 or cnt = 0

Input/Output

Mnemonic	Operands	Addressing Modes	Operation
IN* INB* INL*	dst,src	dst: R src: IR,DA	**Input** dst ← src
IND* INDB* INDL*	dst,src,cnt	dst: IR src: IR cnt: R	**Input and Decrement** dst ← src Autodecrement dst address cnt ← cnt − 1
INDR* INDRB* INDRL*	dst,src,cnt	dst: IR src: IR cnt: R	**Input, Decrement, and Repeat** Repeat dst ← src Autodecrement dst address cnt ← cnt − 1 Until cnt = 0
INI* INIB* INIL*	dst,src,cnt	dst: IR src: IR cnt: R	**Input and Increment** dst ← src Autoincrement dst address cnt ← cnt − 1
INIR* INIRB* INIRL*	dst,src,cnt	dst: IR src: IR cnt: R	**Input, Increment, and Repeat** Repeat dst ← src Autoincrement dst address cnt ← cnt − 1 Until cnt = 0
OTDR* OTDRB* OTDRL*	dst,src,cnt	dst: IR src: IR cnt: R	**Output, Decrement, and Repeat** Repeat dst ← src Autodecrement src address cnt ← cnt − 1 Until cnt = 0
OTIR* OTIRB* OTIRL*	dst,src,cnt	dst: IR src: IR cnt: R	**Output, Increment, and Repeat** Repeat dst ← src Autoincrement src address cnt ← cnt − 1 Until cnt = 0

Input/Output (Continued)

Mnemonic	Operands	Addressing Modes	Operation
OUT* **OUTB*** **OUTL***	dst,src	dst: IR,DA src: R	**Output** dst ← src
OUTD* **OUTDB*** **OUTDL***	dst,src,cnt	dst: IR src: IR cnt: R	**Output and Decrement** dst ← src Autodecrement src address cnt ← cnt − 1
OUTI* **OUTIB*** **OUTIL***	dst,src,cnt	dst: IR src: IR cnt: R	**Output and Increment** dst ← src Autoincrement src address cnt ← cnt − 1

CPU Control

Mnemonic	Operands	Addressing Modes	Operation
COMFLG	flags	flags: IM	**Complement Flag**
DI*	ints	ints: IM	**Disable Interrupt**
EI*	ints	ints: IM	**Enable Interrupt**
HALT*			**Halt**
IRET*			**Interrupt Return** PS ← @SP Autoincrement SP
LDCTL*	dst,src	dst: CTLR src: R or dst: R	**Load Control Register** dst ← src
LDCTLB	dst,src	dst: FLGR src: R or dst: R src: FLGR	**Load Flag Byte Register** dst ← src
LDCTLL*	dst,src	dst: CTLRL src: R or dst: R src: CTLRL	**Load Control Register Long** dst ← src
LDND* **LDNDB*** **LDNDL***	dst,src	dst: R src: IR,EAM or dst: IR,EAM src: R	**Load Normal Data Address Space** dst ← src

*Privileged instruction

CPU Control (Continued)

Mnemonic	Operands	Addressing Modes	Operation
LDNI* **LDNIB*** **LDNIL***	dst,src	dst: R src: IR,EAM or dst: IR,EAM src: R	**Load Normal Instruction Address Space** dst ← src
LDPS*	src	src: IR,EAM	**Load Program Status** PS ← src
LDND* **LDNI*** **LDSD*** **LDSI***	dst,src	dst: R src: IR,EAM	**Load Physical Address** dst ← Physical_Address (src)
NOP			**No Operation**
PCACHE*			**Purge Cache**
PTLB*			**Purge TLB**
PTLBEND* **PTLBENI** **PTLBESD*** **PTLBESI***	src	src: IR,EAM	**Purge TLB Entry**
PTLBN*			**Purge TLB Normal**
RESFLG	flag	flag: IM	**Reset Flag**
SETFLG	flag	flag: IM	**Set Flag**

Bit Field

Mnemonic	Operands	Addressing Modes	Operation
EXTR	dst,src, pos,siz	dst: R src: R,IR,EAM pos: IM,R siz: IM,R	**Extract Field** dst ← src (pos,siz)
EXTRU	dst,src, pos,siz	dst: R src: R,IR,EAM pos: IM,R siz: IM,R	**Extract Unsigned Field** dst ← src (pos,siz)
INSRT	dst,src, pos,siz	dst: R,IR,EAM src: R pos: IM,R siz: IM,R	**Insert Field** dst ← (pos,siz) ← src

*Privileged instruction.

EXTENDED INSTRUCTIONS

The Z80,000 CPU supports extended instructions through the Zilog Extended Processing Architecture (EPA). The EPA facility allows the operations defined in the Z80,000 architecture to be extended by software or hardware. In particular, floating-point operations are supported by the Z8070 Arithmetic Processing Unit (APU) or by a software package that emulates the APU.

Up to four Extended Processing Units (EPUs) can be included in a Z80,000 CPU system. The CPU and EPU cooperate in execution of EPA instructions. When the CPU encounters an EPA instruction, the instruction is transmitted across the external bus to the appropriate EPU. The CPU then performs transactions on the external bus to transfer data between the EPU and memory or the EPU and CPU. Transfers between the EPU and CPU can involve the CPU general-purpose registers or FCW flag byte. EPU internal operations do not require any data transfers. After the data transfers for the EPU instruction are completed, the CPU can continue processing while the EPU performs the operation. While the EPU is processing an instruction, it can drive the $\overline{\text{EPUBSY}}$ signal to stop the CPU.

The data processing operations performed by the EPU are transparent to the CPU. The EPU can execute floating point operations, decimal arithmetic, specialized operating system functions, signal processing operations, or any other that the system designer chooses. For this reason, no mnemonic is listed for the extended instructions, as the mnemonic will depend on the type of EPU. EPUs designed to speed execution of special purpose operations can provide significant performance improvements. The operation of the EPU can be overlapped with operation of the CPU and other EPUs.

The EPA bit in the Flag and Control Word register indicates whether an EPU is present. If no EPU is present, the CPU traps EPA instructions for software simulation. Thus, the EPA facility can be used even with no external support circuitry. This allows software compatibility between systems, whether or not an EPU is present. The system designer can choose to include an EPU in high-performance systems but not in low-cost systems, and software can be developed using the EPA instructions before an EPU is available.

Extended Instructions

Operation	Operands	Addressing Modes
Load EPU from memory dst ← src (n bytes or words)	dst,src,n	dst: EPU src: IM,IR,EAM* n: IM
Load memory from EPU dst ← src (n bytes or words)	dst,src,n	dst: IR,EAM* src: EPU n: IM
Load EPU from CPU dst ← src (n words or longwords)	dst,src,n	dst: EPU src: R n: IM
Load CPU from EPU dst ← src (n words or longwords)	dst,src,n	dst: R src: EPU n: IM
Load EPU from Flags dst ← src	dst,src	dst: EPU src: FLGR
Load Flags from EPU dst ← src	dst,src	dst: FLGR src: EPU
EPU Internal Operation		

*Compact mode allows BX with no displacement for EAM.

CACHE

The CPU implements a Cache mechanism to keep on-chip copies of the most recently referenced memory locations (Figure 10). The CPU examines the cache on memory fetches to determine if the addressed data are located in the cache. If the information is in the cache (a hit), then the CPU fetches from the cache, and no transaction is necessary on the external interface. If the information is not in the cache (a miss), then the CPU performs a Memory Read transaction to fetch the missing information.

The cache stores data in blocks of 16 bytes. Each data word in the cache has an associated validity bit to indicate whether or not the word is a valid copy of the corresponding main memory location. The cache contains 16 blocks, providing 256 bytes of storage.

The cache is fully associative, so that a block currently needed and missing in the cache can replace any block in the cache. Moreover, when a block miss occurs, the least recently used (LRU) block in the cache is replaced. When a cache miss occurs on an instruction fetch, the CPU fetches the missing instruction from memory and prefetches the following words in the block using a Burst transaction. When a cache miss occurs on an operand fetch, the CPU fetches the missing data from memory. (The CPU uses Burst transactions only for fetching operands when more than one data transfer is necessary: longword operands on a 16-bit bus, unaligned operands, string instructions, Load Multiple instructions, and loading Program Status.)

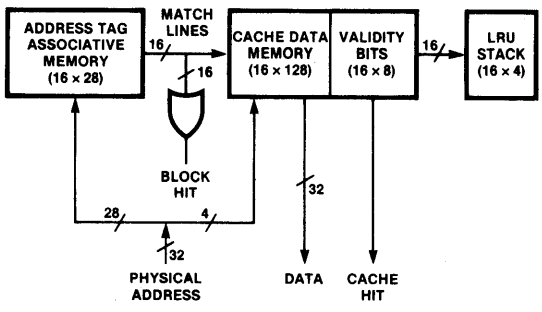


Figure 10. Cache Organization

On store references, the data is written to memory (store through), and if the reference hits in the cache, the data is also written to the cache. If the store reference misses in the cache, the cache is unaffected.

Software has some control over the cache. The cache can be selectively enabled for instruction and data references by bits CI and CD in the SCCL control register. The memory management mechanism allows cacheing to be inhibited for individual pages. The Pcache instruction can be used to invalidate all information in the cache.

The cache has an option, controlled by bit CR in SCCL, to inhibit block replacement on a miss. This option can be used to lock fixed locations into the cache for fast, on-chip access. To do this, the cache is first enabled for block replacement of data references only. Selected blocks are read into the cache. The block replacement algorithm is then disabled, while the cache is enabled for instruction and data references.

PIN DESCRIPTIONS

The CPU has 58 signal lines. Pin functions are shown in Figure 11.

AD_0–AD_{31}. *Address/Data (Bidirectional, active High, 3-state).* These 32 lines are time-multiplexed to transfer address and data. At the beginning of each transaction the lines are driven with the 32-bit address. After the address has been driven, the lines are used to transfer one or more bytes, words, or longwords of data.

$\overline{AS}$. *Address Strobe (Output, active Low, 3-state).* The rising edge of $\overline{AS}$ indicates the beginning of a transaction and shows that the address, ST_0–ST_3, R/$\overline{W}$, BL/$\overline{W}$, BW/$\overline{L}$, N/$\overline{S}$, and $\overline{BRST}$ are valid.

$\overline{BUSREQ}$. *Bus Request (Input, active Low).* A Low on this line indicates that a bus requestor has obtained or is trying to obtain control of the local bus.

$\overline{BUSACK}$. *Bus Acknowledge (Output, active Low).* A Low on this line indicates that the CPU has relinquished control of the local bus in response to a bus request.

$\overline{BRST}$. *Burst (Output, active Low, 3-state).* A Low on this line indicates that the CPU is performing a burst transfer; i.e, multiple Data Strobes following a single Address Strobe.

$\overline{\text{BRSTA}}$. *Burst Acknowledge (Input, active Low).* A Low on this line indicates that the responding device can support burst transfers.

BL/$\overline{\text{W}}$; BW/$\overline{\text{L}}$. *Byte, Longword/Word; Byte, Word/Longword (Output, 3-state).* These two lines specify the data transfer size.

BL/$\overline{\text{W}}$	BW/$\overline{\text{L}}$	Size
High	High	Byte
Low	High	Word
High	Low	Longword
Low	Low	Reserved

$\overline{\text{DS}}$. *Data Strobe (Output, active Low, 3-state).* $\overline{\text{DS}}$ is used for timing data transfers.

$\overline{\text{EPUBSY}}$. *EPU Busy (Input, active Low).* A Low on this line indicates that an EPU is busy. This line is used to synchronize the operation of the CPU with an EPU during execution of an EPA instruction.

$\overline{\text{GREQ}}$. *Global Request (Output, active Low, 3-state).* A Low on this line indicates the CPU has obtained or is trying to obtain control of a global bus.

$\overline{\text{GACK}}$. *Global Acknowledge (Input, active Low).* A Low on this line indicates the CPU has been granted control of a global bus.

$\overline{\text{IE}}$. *Input Enable (Output, active Low, 3-state).* A Low on this line can be used to enable buffers on the AD lines to drive toward the CPU.

$\overline{\text{NMI}}$. *Non-Maskable Interrupt (Input, Edge activated).* A High-to-Low transition on this line requests a non-maskable interrupt.

$\overline{\text{NVI}}$. *Non-Vectored Interrupt (Input, active Low).* A Low on this line requests a non-vectored interrupt.

N/$\overline{\text{S}}$. *Normal/System Mode (Output, Low* = System Mode, 3-state). This line indicates whether the CPU is in Normal or System mode.

$\overline{\text{OE}}$. *Output Enable (Output, active Low, 3-state).* A Low on this line can be used to enable buffers on the AD lines to drive away from the CPU.

CLK. *Clock (Input).* This line is the clock used to generate all CPU timing.

R/$\overline{\text{W}}$. *Read/Write (Output, Low* = Write, 3-state). This signal indicates the direction of data transfer.

$\overline{\text{RESET}}$. *Reset (Input, active Low).* A Low on this line resets the CPU.

RSP_0–RSP_1. *Response (Input).* These lines encode the response to transactions initiated by the CPU. Note that RSP_0 and RSP_1 can be connected together for Z-BUS $\overline{\text{WAIT}}$ timing.

RSP_0	RSP_1	Response
High	High	Ready
Low	High	Bus Error
High	Low	Bus Retry
Low	Low	Wait

ST_0–ST_3. *Status (Output, active High, 3-state).* These lines specify the kind of transaction occurring on the bus. (See Table 4.)

$\overline{\text{VI}}$. *Vectored Interrupt (Input, active Low).* A Low on this line requests a vectored interrupt.

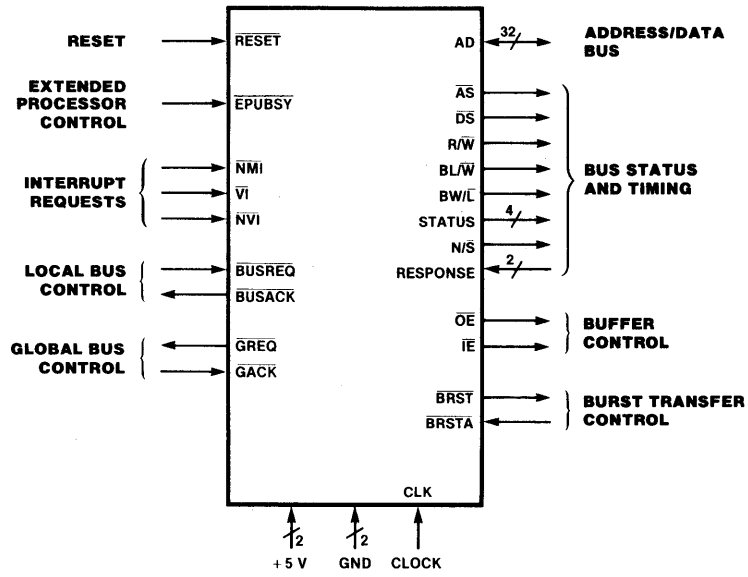


Figure 11. Pin Functions

MULTIPROCESSOR CONFIGURATIONS

The CPU provides support for interconnection in four types of multiprocessor configurations (Figure 12): coprocessor, slave processor, tightly-coupled multiple CPUs, and loosely-coupled multiple CPUs.

Coprocessors, such as the Z8070 Arithmetic Processing Unit, work synchronously with the CPU to execute a single instruction stream using the Extended Processing Architecture facility. The signal $\overline{EPUBSY}$ is dedicated for connection with coprocessors.

Slave processors, such as the Z8016 DMA Transfer Controller, perform dedicated functions asynchronously to the CPU. The CPU and slave processor share a local bus, of which the CPU is the default master, using the CPU's $\overline{BUSREQ}$ and $\overline{BUSACK}$ lines.

Tightly-coupled, multiple CPUs execute independent instruction streams and generally communicate through shared memory located on a common (global) bus using the CPU's $\overline{GREQ}$ and $\overline{GACK}$ lines. Each CPU is default master of its local bus, but the global bus master is chosen by an external arbiter. The CPU also provides status information about interlocked memory references (for Test and Set, Increment Interlocked, and Decrement Interlocked instructions), which can be used with multi-ported memories.

Loosely-coupled, multiple CPUs generally communicate through a multiple-ported peripheral, such as the Z8038 FIO. The Z80,000 CPU's I/O and interrupt facilities can support loosely-coupled multiprocessing.

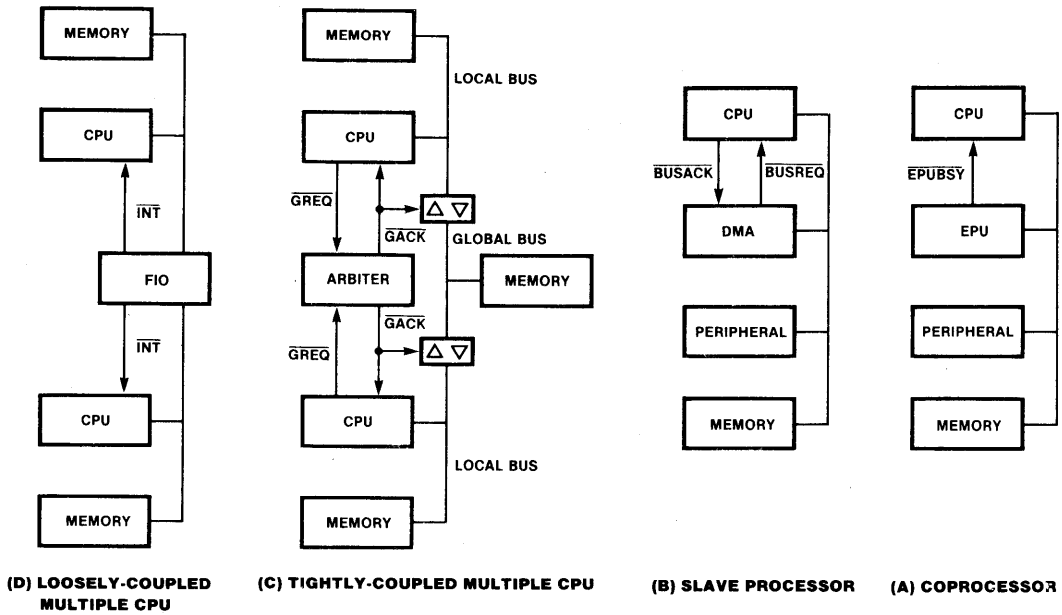


Figure 12. Multiprocessor Configurations

HARDWARE INTERFACE CONTROL REGISTER

The Hardware Interface Control register (HICR) specifies certain characteristics of the hardware configuration surrounding the CPU, including bus speed, memory data path width, and number of automatic Wait states. The physical memory address space is divided into two sections, M_0 and M_1, selected by bit 30 of the address. A typical system would locate slow, 16-bit wide bootstrap ROM in M_0 and faster 32-bit wide dynamic RAM in M_1. The physical I/O address space is similarly divided into two sections, I/O_0 and I/O_1, selected by bit 30 of the address.

Fields in HICR specify the following interface characteristics (see Figure 3):

Bus speed (S)—The bus clock frequency is either 1/2 or 1/4 the clock frequency.

Memory data path (M_0.DP, M_1.DP)—The data path width for M_0 and M_1 are each specified as 16 or 32 bits.

Automatic Wait states (M_0.W, M_1.W, I/O_0.W, I/O_1.W, IACK.W1, IACK.W2)—The number of Wait states automatically inserted by the CPU for references to M_0, M_1, I/O_0, I/O_1, and Interrupt Acknowledge, are separately specified.

Global bus protocol control (LAD, GE)—The CPU can access a global bus (a bus shared with other CPUs). On references to the global bus, the CPU must use a re-

quest/acknowledge handshake with an external arbiter. The GE field enables the use of the global bus; the LAD field selects the portion of the address space used for references to the global bus.

Minimum Address Strobe rate (MASR)—This optional feature ensures that an Address Strobe will be generated at least once every 16 bus clock cycles. This is useful for refreshing pseudostatic RAMS.

EPU overlap (EPUO)—This bit, along with another bit in an EPU control register, controls the degree of overlap for CPU and EPU operations. The degree of overlap can be limited to simplify debugging and recovery from exceptions, although to do so reduces overall execution speed.

CPU TIMING

The CPU performs transactions on the external interface to transfer data for fetching instructions, fetching and storing operands, processing exceptions, and performing memory management. In addition, the CPU performs Internal Operation and Halt transactions, which do not transfer data. Each transaction occurs during a sequence of bus clock cycles, named T_1, T_2, etc.

The CPU has a single clock line, CLK, used to generate all timing. Internally, the CPU derives another clock for bus timing by dividing CLK by 2 or 4. The scale factor for bus timing (2 or 4) is selected at Reset. In the AC timing characteristics for the CPU, input setup and hold times and output delays are specified with respect to a rising edge of CLK. When CPU output transitions occur on different clock edges, the time between the transitions is specified in terms of a constant delay and a variable number of CLK cycles. The number of CLK cycles depends on the bus timing scale factor, type of transaction, and number of Wait states.

In the logical timing diagrams that follow, the signal transitions on the bus are shown in relation to the bus clock, BCLK. The beginning of a transaction, signified by a falling edge of $\overline{AS}$, always occurs on a rising edge of BCLK. The BCLK signal is derived internally to the CPU as

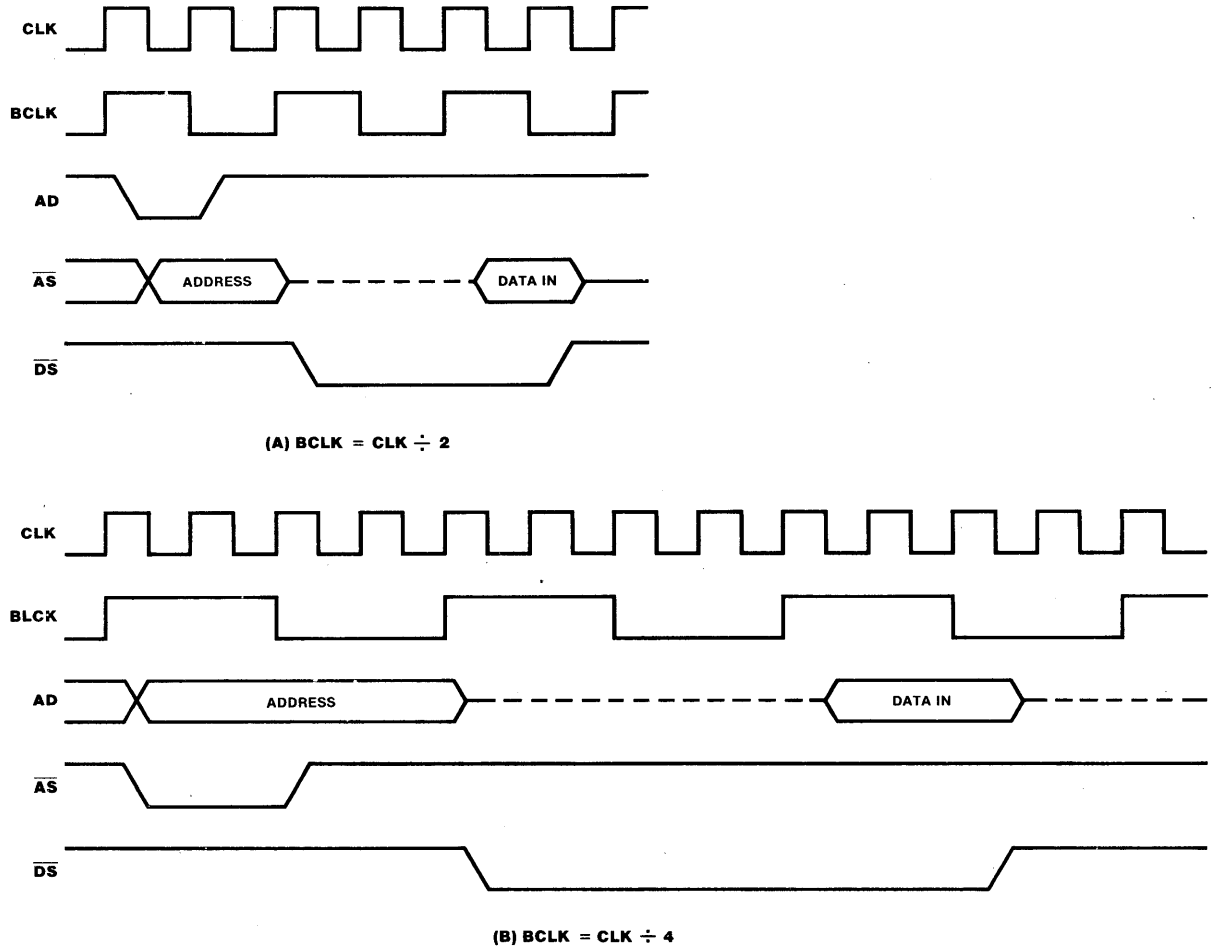


Figure 13. Memory Read Timing for Different Bus Scale Factors

described above, but is not available on the pins. BCLK can also be derived externally to the CPU by dividing CLK by the selected bus timing scale factor. (The Reset section discusses synchronization of the internal and external bus clocks). The timing diagrams in Figure 13 show example memory read transactions using the different scale factors.

In the description of bus transactions that follow, the term "asserted" means an active signal and "negated" means an inactive signal. A signal is either active when High or when Low, as specified in the pin functions.

BUS TRANSACTIONS

All bus transactions begin with Address Strobe ($\overline{AS}$) asserted and then negated. On the rising edge of $\overline{AS}$ the lines for status (ST_0–ST_3), Read/Write (R/$\overline{W}$), data transfer size (BW/$\overline{L}$, BL/$\overline{W}$), and Normal/System (N/$\overline{S}$) are valid. The status lines indicate the type of transaction being initiated (Table 4). The R/$\overline{W}$ line indicates the direction of data transfer. The data transfer size indicates whether a byte, word, or longword of data is being transferred. The N/$\overline{S}$ line indicates the CPU's operating mode. The following sections describe timing for the different transactions.

Table 4. Status Codes

ST_3-ST_0	Definition
0 0 0 0	internal operation
0 0 0 1	CPU-EPU (data)
0 0 1 0	I/O
0 0 1 1	halt
0 1 0 0	CPU-EPU (instruction)
0 1 0 1	$\overline{NMI}$ acknowledge
0 1 1 0	$\overline{NVI}$ acknowledge
0 1 1 1	$\overline{VI}$ acknowledge
1 0 0 0	cacheable CPU-memory (data)
1 0 0 1	non-cacheable CPU-memory (data)
1 0 1 0	cacheable EPU-memory
1 0 1 1	non-cacheable EPU-memory
1 1 0 0	cacheable CPU-memory (instruction)
1 1 0 1	non-cacheable CPU-memory (instruction)
1 1 1 0	reserved
1 1 1 1	interlocked CPU-memory (data)

On the rising edge of $\overline{AS}$, the address on the AD lines is also valid. Addresses are not required and therefore are undefined for Internal Operation, Halt, Interrupt Acknowledge, and CPU-EPU data transactions.

The CPU uses Data Strobe ($\overline{DS}$) to time the data transfer. (Internal Operation and Halt transactions do not transfer data, and thus do not assert $\overline{DS}$.) For Write operations (R/$\overline{W}$ = Low) the CPU asserts $\overline{DS}$ when valid data is on the AD lines. For Read operations (R/$\overline{W}$ = High) the CPU makes the AD lines 3-state before asserting $\overline{DS}$, so the addressed device can put its data on the bus. The CPU samples the data in the middle of a bus cycle, while negating $\overline{DS}$.

The AD lines can be used to transfer bytes, words, or longwords. For Read transactions, the three cases are handled as follows:

- Byte transfers use AD_0–AD_7; AD_8–AD_{31} are ignored.
- Word transfers use AD_0–AD_{15}; AD_{16}–AD_{31} are ignored.
- Longword transfers use AD_0–AD_{31}.

For Write transactions, the three cases are handled as follows:

- Byte transfers replicate the data on AD_0–AD_7, AD_8–AD_{15}, AD_{16}–AD_{23}, and AD_{24}–AD_{31}.
- Word transfers replicate the data on AD_0–AD_{15} and AD_{16}–AD_{31}.
- Longword transfers use AD_0–AD_{31}.

The signals Input Enable ($\overline{IE}$) and Output Enable ($\overline{OE}$) can be used to enable buffers on the bidirectional AD lines. $\overline{IE}$ is asserted when the buffers drive toward the CPU; $\overline{OE}$ is asserted when the buffers drive away from the CPU. Whenever the direction for the AD lines changes, neither $\overline{IE}$ nor $\overline{OE}$ is asserted for at least one CLK cycle.

To transfer more than one data item, the CPU can perform Burst transactions. The data items are transferred in the same direction, and must be equal in size. Data Strobe is used to time each transfer. The CPU asserts Burst ($\overline{BRST}$) to indicate a burst transfer. The responding device asserts Burst Acknowledge ($\overline{BRSTA}$) if it is capable of supporting burst transfers. If $\overline{BRSTA}$ is not asserted, the CPU transfers only a single data item.

RESPONSE

Any time data is transferred, the responding device returns a code on the Response lines (RSP_0–RSP_1) to indicate Ready, Wait, Bus Error, or Bus Retry. The response is sampled at a time specific to each type of transaction, generally before the AD lines are sampled or $\overline{DS}$ is negated.

Ready indicates the completion of a successful transfer.

Wait indicates that the responding device needs more time to complete the transaction. The CPU waits one bus cycle before sampling the response again to accommodate slow memory or peripherals.

Bus Error indicates that a fatal error has occurred during the transaction; for example, bus timeout for a nonexistent device. Bus Error is treated as an exception by the CPU.

Bus Retry indicates that the transaction should be tried again; for example, a transient parity error is detected. The CPU tries the transaction again.

The CPU can insert Wait states automatically under control of several fields in the Hardware Interface Control Register. If an automatic Wait state is programmed for a bus cycle, the CPU ignores the response and Wait is assumed. Thus, Wait states can be inserted automatically by the CPU or upon request of the responding device.

It must be emphasized that the RSP_0–RSP_1 lines are synchronous. Thus, they must meet the specified setup and hold times for correct operation. A simple system using only Z-BUS $\overline{WAIT}$ can be implemented by connecting $\overline{WAIT}$ to RSP_0 and RSP_1.

CPU-MEMORY TRANSACTIONS

The CPU uses status 1000, 1001, 1100, 1101, or 1111 to read from and write to memory. The transactions involve a single data transfer or multiple, burst data transfers.

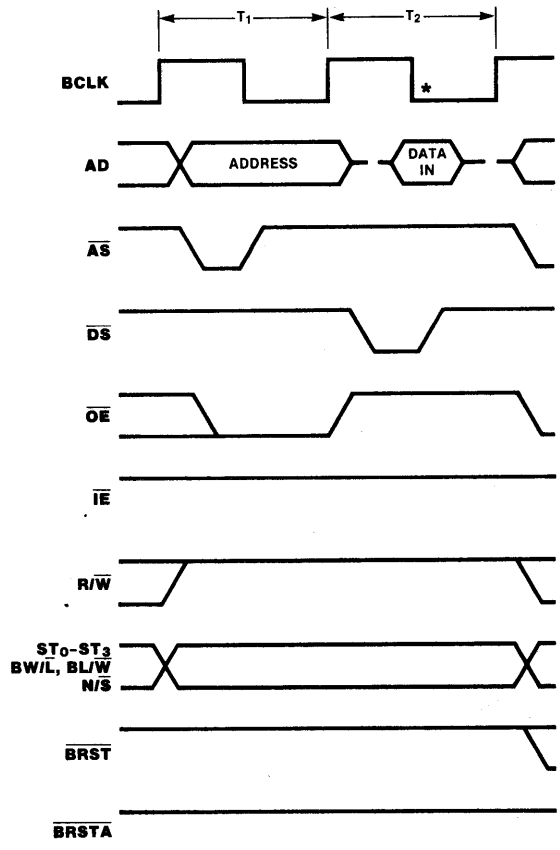
Single Memory Read

Figure 14 shows timing for a single memory read transaction with no Wait states. $\overline{AS}$ is asserted during the first half of T_1. The rising edge of $\overline{AS}$ indicates that the address on AD and control signals ST_0–ST_3, R/$\overline{W}$, BW/$\overline{L}$, BL/$\overline{W}$, and N/$\overline{S}$ are valid. The control signals remain valid for the duration of the transaction. $\overline{BRST}$ is negated during the transaction because only a single data item is being transferred. At the beginning of T_2 the CPU stops driving the address, asserts $\overline{DS}$, and prepares to receive data from memory. In the middle of T_2 RSP_0–RSP_1 are sampled Ready, the input data is latched, and $\overline{DS}$ is negated. The signal $\overline{OE}$ is asserted during T_1; however, for this two-cycle read transaction, $\overline{IE}$ is not asserted.

The CPU can insert Wait states in the middle of T_2 if RSP_0–RSP_1 are sampled Wait or if automatic Wait states are programmed in the appropriate field of HICR. The duration of a Wait state is one BCLK cycle.

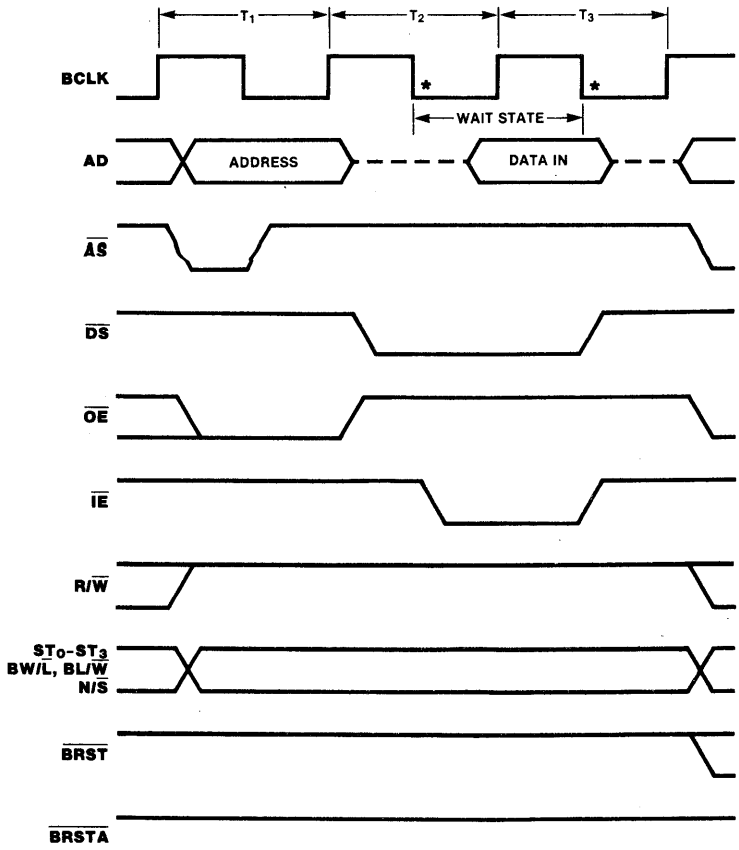
The timing for a single memory Read transaction with one Wait state is shown in Figure 15. This is not a true Wait state because $\overline{IE}$ is asserted in the middle of T_2 and continues until the middle of T_3. For memory Read transactions longer than two bus cycles, either because of Wait states or Burst transfers, $\overline{IE}$ is asserted from the middle of T_2 until the end of data transfer. The signals $\overline{OE}$ and $\overline{IE}$ can be used to control buffers on the AD lines.

For memory Read transactions, the data transfer size is equal to the data path width specified in HICR. The memory should transfer the aligned longword addressed by AD_2–AD_{31} (ignoring AD_0–AD_1) for a 32-bit data path or the aligned word addressed by AD_1–AD_{31} (ignoring AD_0) for a 16-bit data path. The CPU selects the required bytes from the transferred word or longword.



*RSP_0–RSP_1 and data sampled.

Figure 14. Single Memory Read Timing



*RSP_0-RSP_1 and data sampled.

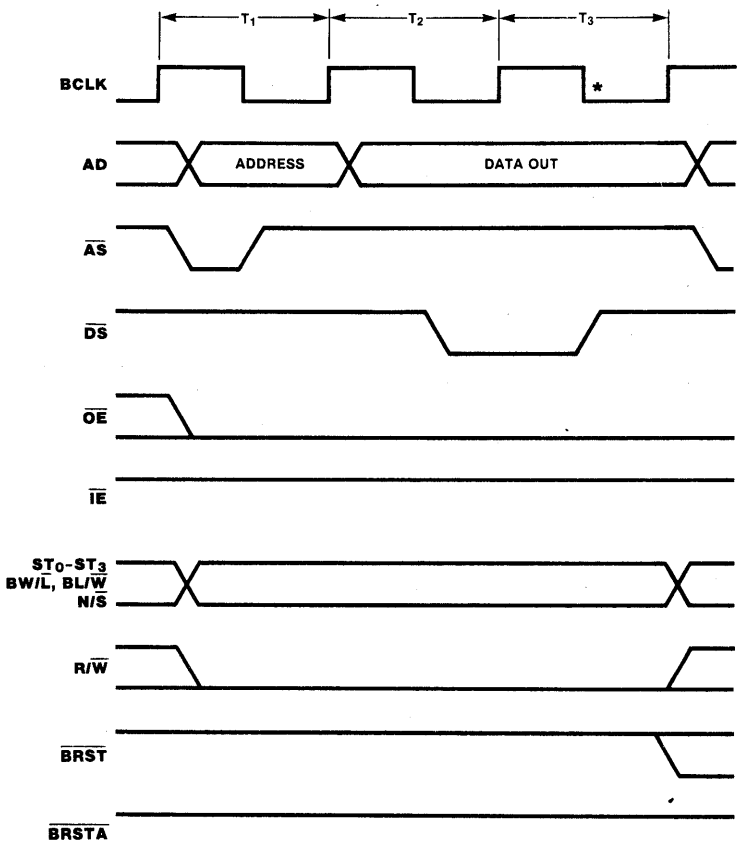
Figure 15. Single Memory Read Timing (1 Wait State)

Single Memory Write

A single memory Write transaction (Figure 16) begins with $\overline{AS}$ to indicate that address and control signals are valid. At the beginning of T_2, the CPU stops driving the address and starts driving the data. In the middle of T_2, $\overline{DS}$ is asserted. The CPU negates $\overline{DS}$ in the middle of T_3. $\overline{OE}$ is asserted beginning at T_1 and continues for the duration of the transaction. The CPU samples RSP_0–RSP_1 in the middle of T_3.

For memory Write transactions, the data transfer size is less than or equal to the data path width specified in HICR. Bytes and words can be written to a 16-bit memory; bytes, words, and longwords can be written to a 32-bit memory. The CPU writes bytes to any address, but words and longwords are always written to an aligned address (i.e., words are always written to an even address and longwords are always written to an address that is a multiple of four). When a program writes a word or longword to an unaligned address, the CPU performs two or more write transactions to aligned addresses. For example, if the program writes a word to an odd address, the CPU first writes the more significant byte to the odd address, then it writes the less significant byte to the successive even address.

Single memory read and write timing differ slightly from Z-BUS specifications. The minimum Read transaction is two bus cycles, and the slave response is sampled at the end of the data transfer. For the Z-BUS, the minimum Read transaction is three cycles, and the slave response is sampled one cycle before the end of the data transfer. For strict Z-BUS compatibility, it is possible to program one automatic Wait state for memory read and to delay the slave response with an external flipflop.



*RSP_0–RSP_1 sampled.

Figure 16. Single Memory Write Timing

Burst Memory Transactions

Burst memory transactions use multiple Data Strobes following a single Address Strobe to transfer data at consecutive memory addresses. The signals $\overline{BRST}$ and $\overline{BRSTA}$ control the Burst transaction. The CPU uses Burst transactions to prefetch a cache block on an instruction fetch cache miss. The CPU also uses Burst transactions to fetch or store operands when more than one transfer is necessary, as with unaligned operands, string instructions, Load Multiple instructions, and loading of program status.

If the memory does not support Burst transfers, the Burst transfer protocol described below (Figure 17) allows $\overline{BRSTA}$ to be tied High. The CPU then separates the Burst transaction into a sequence of single transfers.

At the beginning of a Burst transaction, the CPU asserts $\overline{BRST}$ along with other control signals. When the CPU continues to assert $\overline{BRST}$ at the falling edge of $\overline{DS}$, this indicates to memory that the CPU can support another data transfer following the one in process. When the CPU negates $\overline{BRST}$ before the falling edge of $\overline{DS}$, this indicates to memory that the current transfer is the last in the transaction.

When $\overline{BRSTA}$ is asserted at the time the RSP_0–RSP_1 lines are sampled Ready, this indicates to the CPU that memory can support another data transfer following the one in process. When $\overline{BRSTA}$ is negated at the time the RSP_0–RSP_1 lines are sampled Ready, this indicates to the CPU that the current data transfer is the last in the transaction.

The Burst transaction can be terminated by either the CPU or memory. If memory terminates the transfer by negating $\overline{BRSTA}$, the CPU responds by negating $\overline{BRST}$ when $\overline{DS}$ is negated. (See the example for Burst Memory Read.) If the CPU terminates the transfer by negating $\overline{BRST}$ before the falling edge of $\overline{DS}$, memory responds by negating $\overline{BRSTA}$. (See the example for Burst Memory Write.) The CPU terminates the burst transaction when all the required data items have been transferred or after reaching the end of an aligned, 16-byte block.

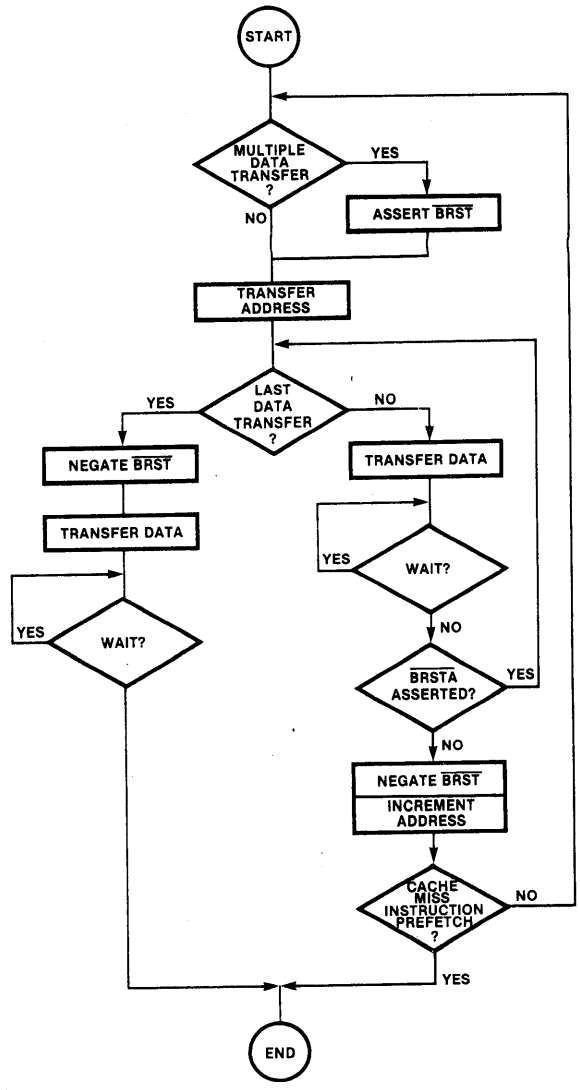
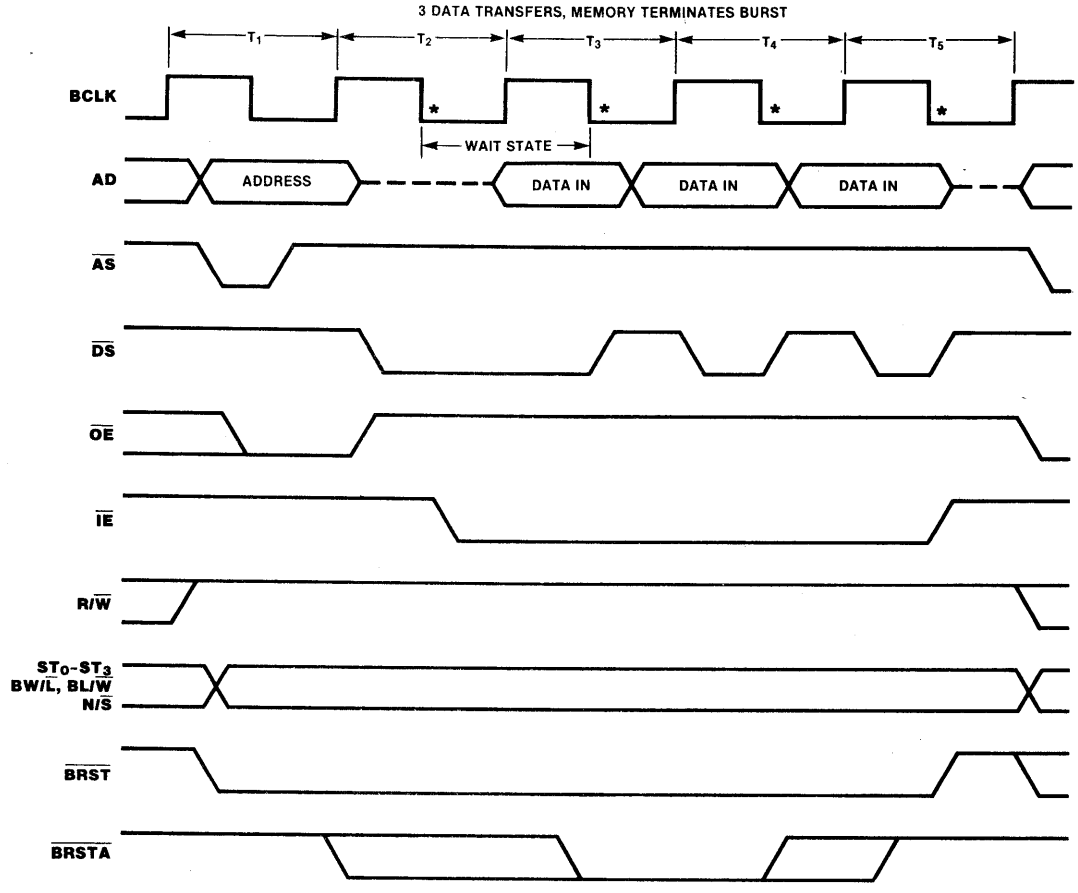


Figure 17. CPU Burst Transfer Protocol

Burst Memory Read

Figure 18 shows timing for a Burst Memory Read transaction with one Wait state. In this example, three data items are transferred, after which memory terminates the burst. $\overline{BRST}$ is asserted at the beginning of T_1; otherwise, the timing for the first transfer is identical to a single memory read. In the middle of T_3 the CPU samples RSP_0–RSP_1 Ready, latches the data, and samples $\overline{BRSTA}$ active. During T_4 the second data item is transferred, accompanied by $\overline{DS}$. The time for the second and subsequent transfers can be extended with wait states if RSP_0–RSP_1 are sampled Wait; the CPU does not insert automatic Wait states after the first transfer.

During T_5 the third data item is transferred. At the same time the RSP_0–RSP_1 lines are sampled Ready, the data is latched and $\overline{BRSTA}$ is sampled inactive. Memory terminates the Burst transfer, and the CPU responds by negating $\overline{BRST}$.



*RSP_0 – RSP_1, $\overline{BRSTA}$, and data sampled.

Figure 18. Burst Memory Read Timing (1 Wait State)

Burst Memory Write

Figure 19 shows timing for a Burst Memory Write transaction with no Wait states. In this example, two data items are transferred, and the CPU terminates the burst. $\overline{BRST}$ is asserted at the beginning of T_1; otherwise, the timing for the first transfer is identical to a single memory write. In the middle of T_3 the CPU samples RSP_0–RSP_1 Ready and $\overline{BRSTA}$ active. At the beginning of T_4 the CPU negates $\overline{BRST}$ indicating that one more data transfer will follow. During T_4 the second data item is transferred, accompanied by $\overline{DS}$. The time for the second and subsequent transfers can be extended with Wait states if RSP_0–RSP_1 are sampled Wait; the CPU does not insert automatic Wait states after the first transfer. Memory recognizes that the CPU has terminated the burst transfer and responds by negating $\overline{BRSTA}$ before the end of T_4.

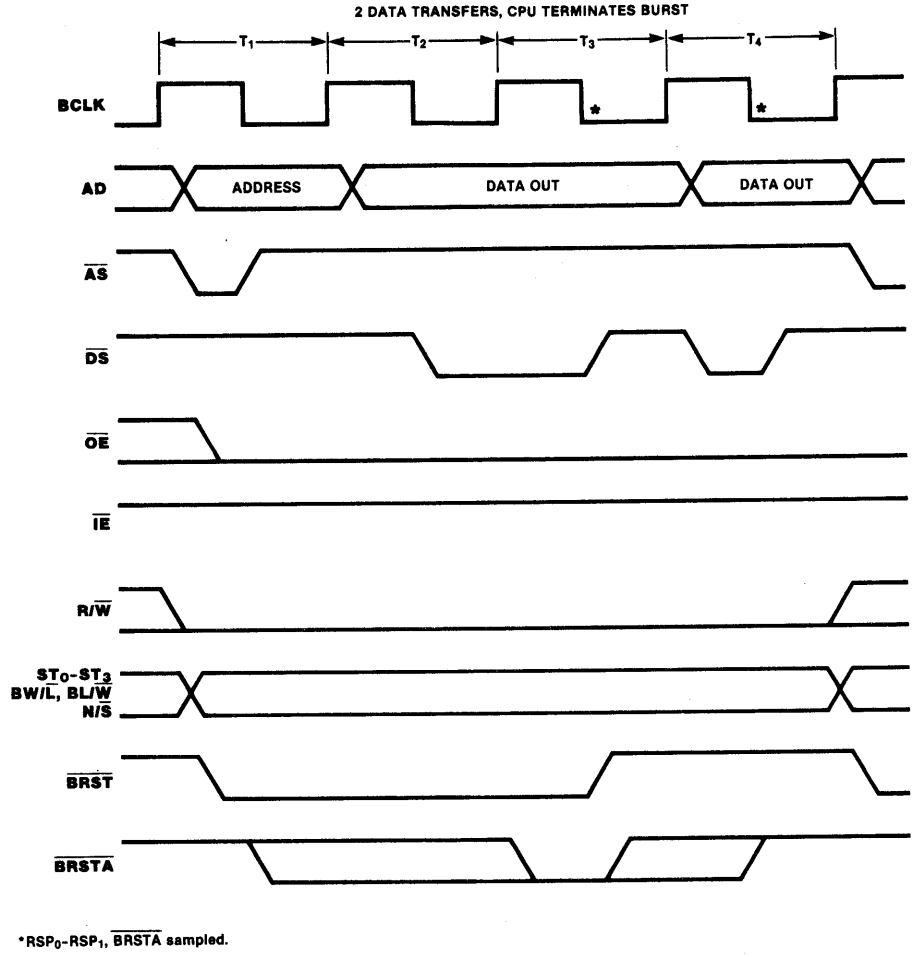


Figure 19. Burst Memory Write Timing

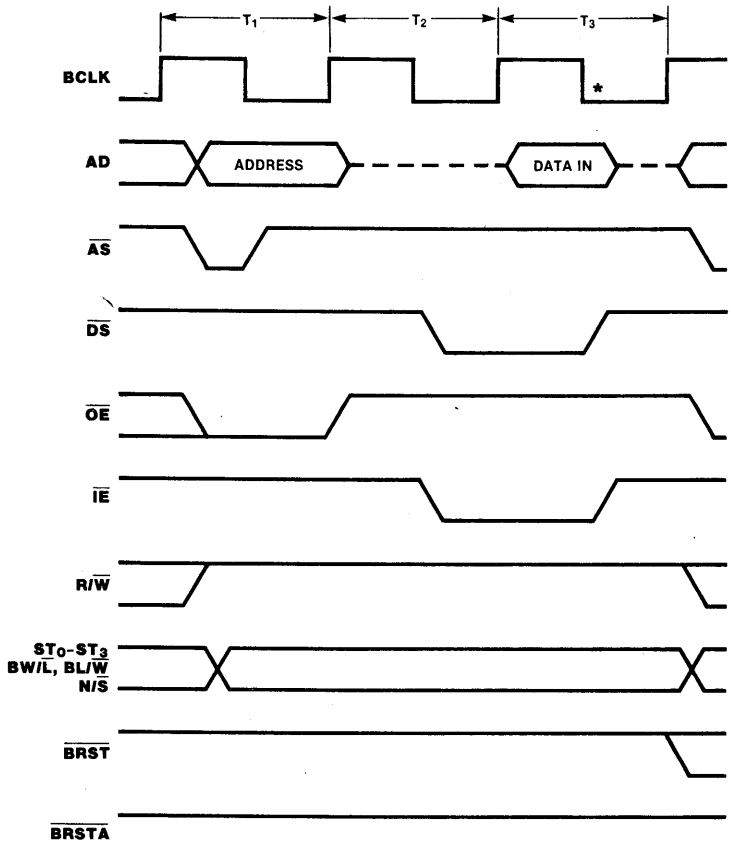
Interlocked Memory Transactions

In some multiprocessing configurations, the CPU must inhibit other bus masters from referencing shared memory while the CPU performs two or more transactions. The CPU does this by using status 1111 with any of the memory transactions described above. A status of 1111 indicates a non-cacheable data reference with interlock protection. The CPU uses interlock protection for data references associated with Test and Set, Decrement Interlocked, and Increment Interlocked instructions. The CPU also uses interlock protection for references to address translation table entries when loading the Translation Lookaside Buffer.

INPUT/OUTPUT TRANSACTIONS

The CPU uses status 0010 to read from and write to I/O ports. I/O transactions are generated by I/O instructions and, when address translation is enabled, by data references to pages with bit 31 of the page table entry set to 1.

The timing for I/O and memory transactions is very similar. The major difference is that $\overline{DS}$ falls in the middle of T_2 for I/O read timing, compared to the beginning of T_2 for memory read timing. This allows peripheral devices more time for address decoding. Another difference is that the data transfer size (byte, word, or longword) for I/O transactions is specified by the instruction, not by HICR. The final difference is that the CPU does not support Burst I/O transactions. Figure 20 shows timing for an I/O Read transaction. Single I/O Write timing is the same as that shown for a single memory write (Figure 16).



*RSP_0-RSP_1 and data sampled.

Figure 20. I/O Read Timing

EPU TRANSACTIONS

The CPU and EPU cooperate in the execution of EPA instructions (Figure 21). When the CPU encounters an EPA instruction and the EPA bit in the FCW is 1, the CPU broadcasts the first two words of the instruction to the EPUs in the system using the CPU-EPU instruction transfer transaction. All EPUs in the system recognize the transaction, but only one of four possible EPUs is selected by bits 16 and 17 of the EPU instruction. The CPU also transfers the PC value for the instruction, which the selected EPU saves for use in exception handling. If data transfers are required to complete the instruction, the CPU controls the data transfer transactions while the EPU drives or receives the data.

The signal $\overline{\text{EPUBUSY}}$, output from the EPU, is used to synchronize the CPU and EPU in executing EPA instructions. The CPU must sample $\overline{\text{EPUBSY}}$ inactive before initiating an EPU instruction transfer. If data transfers are required, the CPU must sample $\overline{\text{EPUBSY}}$ inactive before initiating the first transfer. $\overline{\text{EPUBSY}}$ is also used to control the degree of overlap between CPU and EPU instruction execution. Ordinarily, the CPU can continue processing other instructions after performing the data transfers associated with an EPA instruction and before the EPU has completed executing the instruction. To simplify debugging and recovery from exceptions, a Non-Overlap mode is provided, controlled by the EPU 0 bit in HICR. In Non-Overlap mode, the CPU samples $\overline{\text{EPUBSY}}$ in the middle of the bus cycle during which the last data transfer for an EPA instruction occurs. If $\overline{\text{EPUBSY}}$ is asserted, the CPU ceases processing instructions or interrupts until $\overline{\text{EPUBSY}}$ is sampled inactive in the middle of a bus cycle.

To simplify system hardware, the AD lines for CPU and EPU should be wired together with no buffers between them. If the AD lines are separated by buffers, external circuitry must generate $\overline{\text{IE}}$ and $\overline{\text{OE}}$ timing for CPU-to-EPU Data Read and EPU-to-Memory Write transactions.

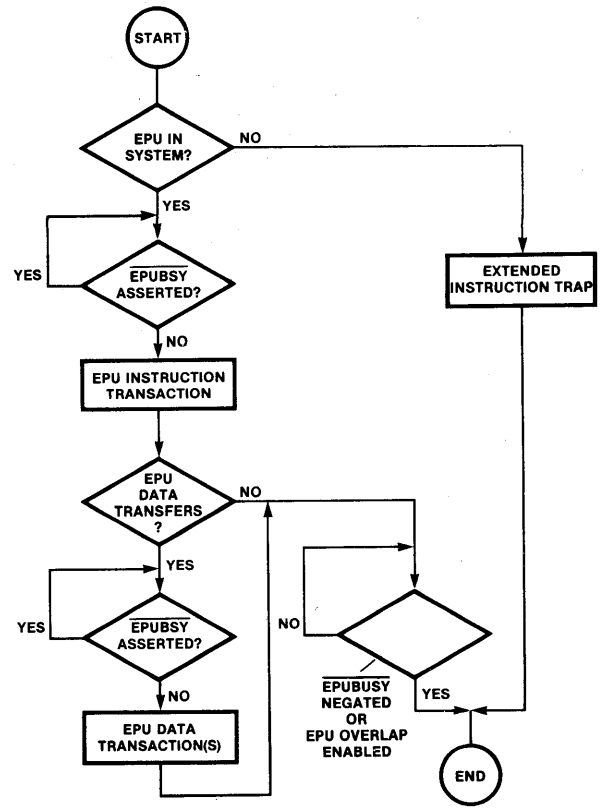
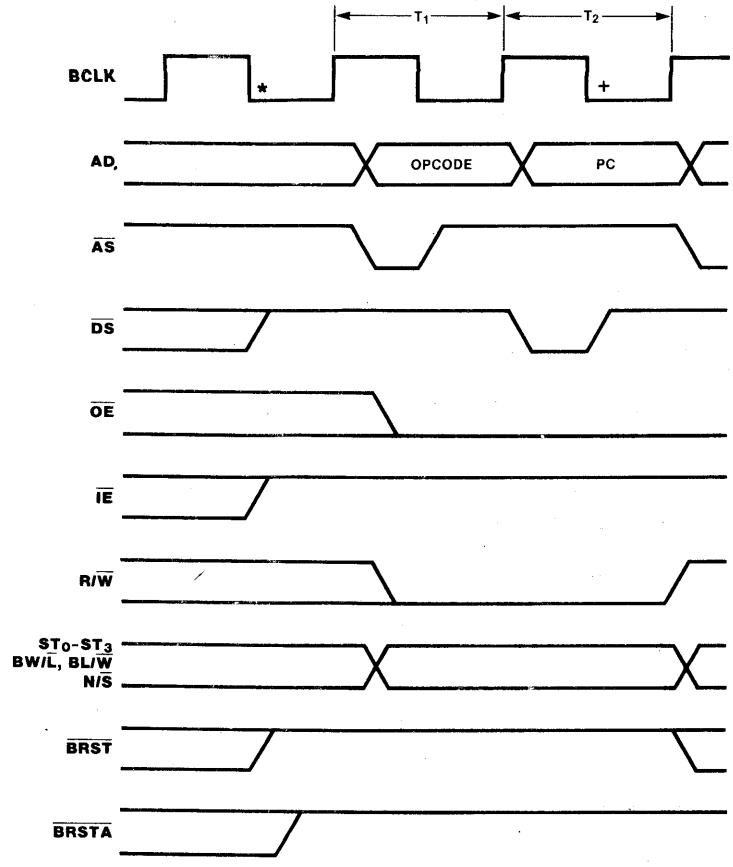


Figure 21. EPA Instruction Processing

CPU-to-EPU Instruction Transfer

Figure 22 shows timing for a CPU-to-EPU Instruction Transfer transaction with status 0100. The rising edge of $\overline{AS}$ indicates that AD lines and status are valid. During T_1, the AD lines are used to transfer the opcode, i.e., the first two words of the EPA instruction. At the beginning of T_2 the CPU stops driving the opcode, asserts $\overline{DS}$, and starts driving PC on the AD lines. The CPU negates $\overline{DS}$ in the middle of T_2. The data transfer size for the transaction is longword. The CPU ignores RSP_0–RSP_1.



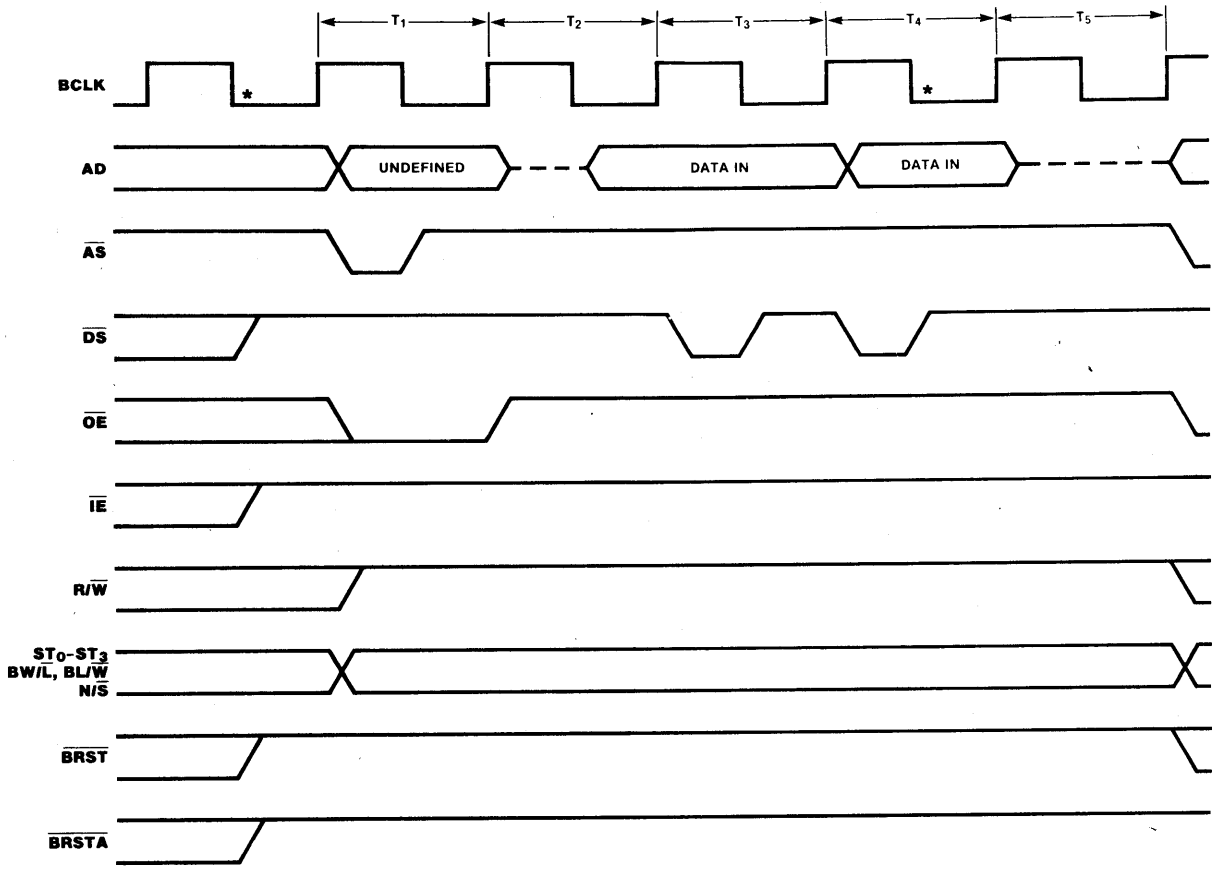
*$\overline{EPUBSY}$ sampled.
+ $\overline{EPUBSY}$ sampled if EPU internal operation.

Figure 22. CPU-to-EPU Instruction Transfer Timing

CPU-to-EPU Data Transfer Transactions

Transactions to transfer data between the CPU and EPU use status 0001. The EPA instruction opcode indicates the number of words to be transferred. One or more longwords of data are transferred until all words have been transferred. If the last transfer contains a single word, the data is replicated on AD_0–AD_{15} and AD_{16}–AD_{31}. The CPU does not assert $\overline{BRST}$ and ignores RSP_0–RSP_1 and $\overline{BRSTA}$.

Figure 23 shows timing for a CPU-to-EPU Data Read transaction. This example has two data transfers; any number of data transfers between one and eight is possible. The rising edge of $\overline{AS}$ indicates that status and control signals are valid. The CPU stops driving the AD lines at the end of T_1, and the EPU begins driving the AD lines in the middle of T_2. At the beginning of T_3, the CPU asserts $\overline{DS}$. In the middle of T_3, the CPU samples the data and negates $\overline{DS}$. The second longword of data is transferred during T_4. After the last data transfer, the CPU inserts an idle bus cycle (T_5 in the example) during which neither the CPU nor EPU drive the AD lines.



*EPUBSY sampled.

Figure 23. CPU-to-EPU Data Read Timing

Figure 24 shows timing for a CPU-to-EPU data write transaction. This example has three data transfers; any number of data transfers between one and eight is possible. Timing for the first transfer is identical to the CPU-to-EPU instruction transfer transaction. A second longword of data is transferred during T_3, and the third longword is transferred during T_4.

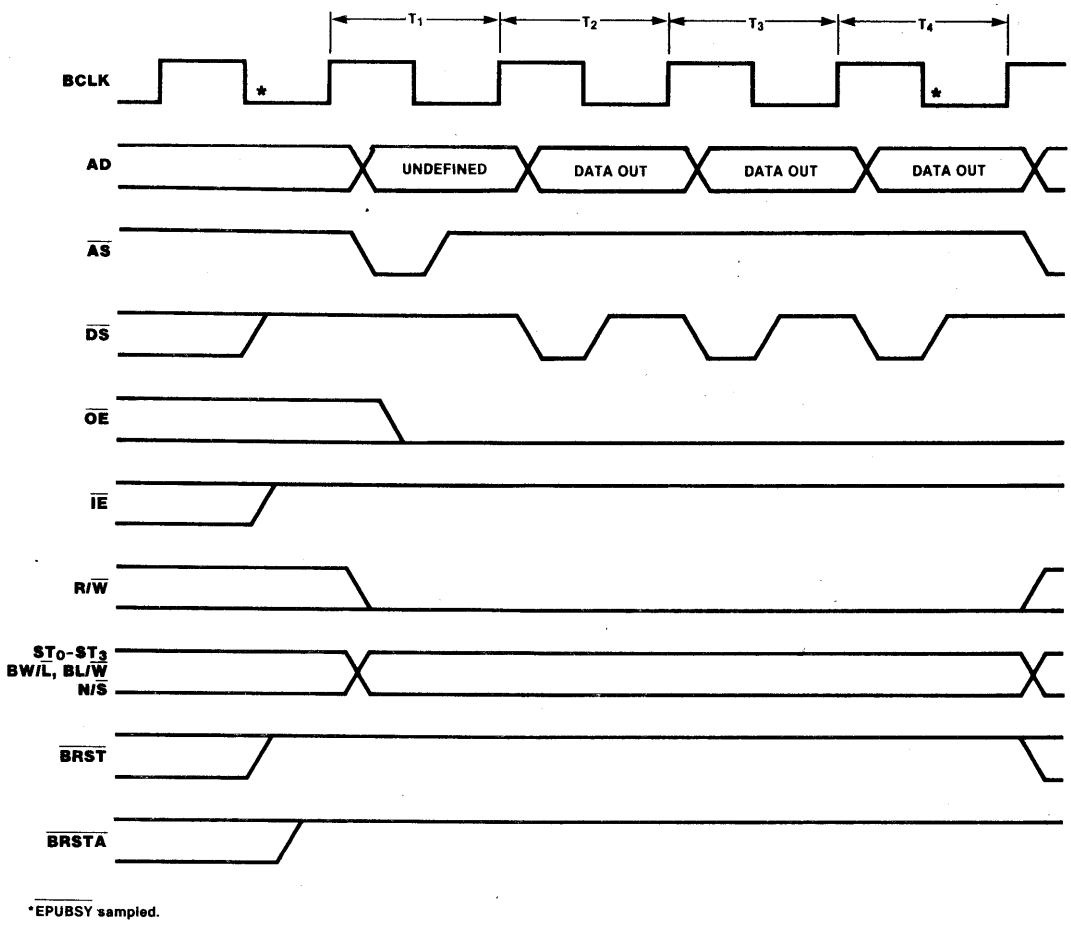
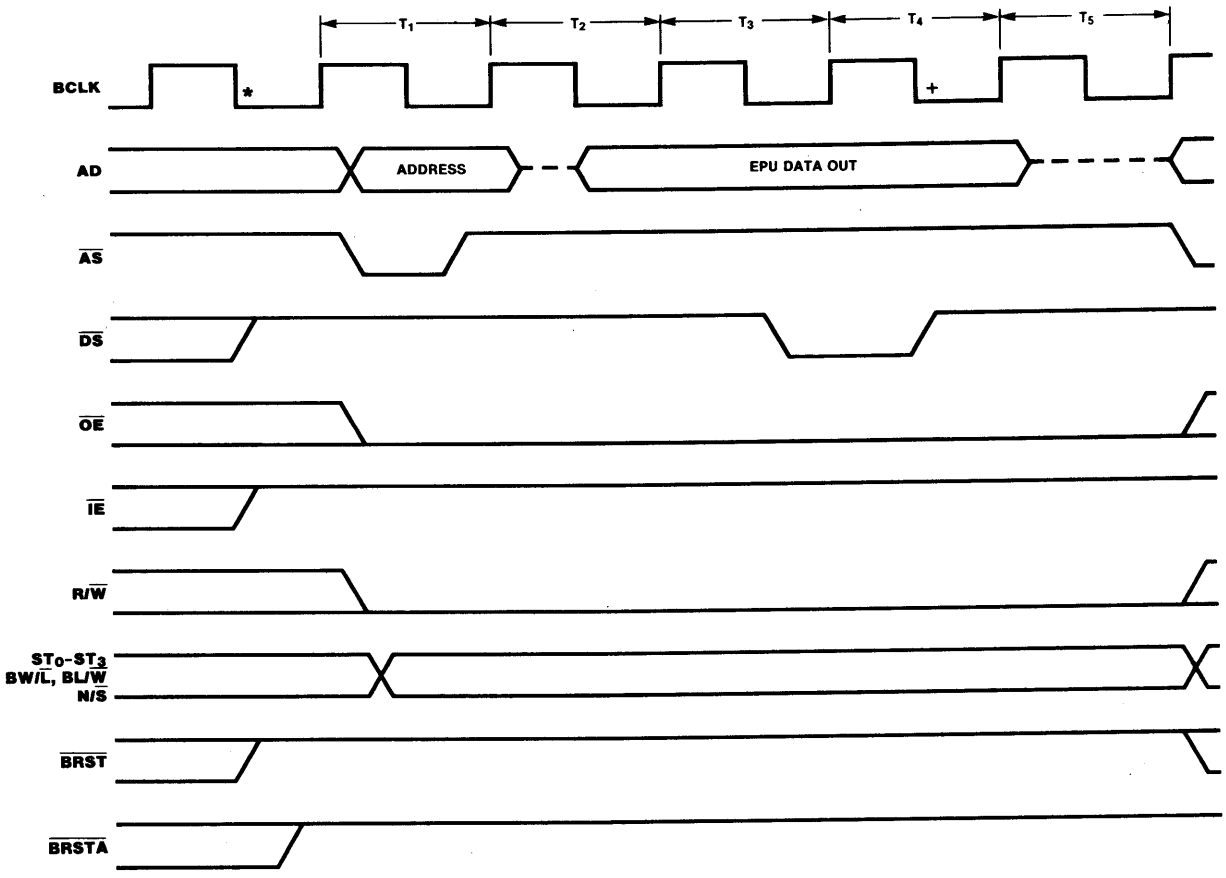


Figure 24. CPU-to-EPU Data Write Timing

EPU-to-Memory Transactions

The CPU uses status 1010 or 1011 for the EPU to read from and write to memory. The timing is identical for EPU-to-memory read and CPU-to-memory read. The EPU monitors the CPU timing on the bus, and uses the two least significant address bits, the data transfer size, and the length of the operand from the instruction to select the bytes it needs from the AD lines.

The timing for an EPU-to-memory write transaction differs slightly from a CPU-to-memory write transaction. Two extra bus cycles are included to pass the AD lines from the CPU to EPU after the address transfer and from EPU back to CPU after the last data transfer. Figure 25 shows an example for a single EPU-to-memory write transaction with no Wait states. The CPU stops driving the AD lines at the end of T_1; the EPU begins driving them in the middle of T_2. $\overline{DS}$ is asserted in the middle of T_3, one bus cycle later than for CPU-to-memory write timing. The CPU negates $\overline{DS}$ in the middle of T_4. The CPU can insert Wait states in the middle of T_4. The EPU continues to drive the AD lines until the end of T_4. After the last data transfer the CPU inserts an idle bus cycle (T_5 in this example) during which neither the CPU nor EPU drive the AD lines. EPU-to-memory burst write transactions are similarly extended by two bus cycles more than with CPU-to-memory burst write timing. One cycle is inserted before the first data transfer, and another after the last data transfer.



*$\overline{EPUBSY}$ sampled.
+ RSP_0-RSP_1 sampled; $\overline{EPUBSY}$ sampled if last transaction.

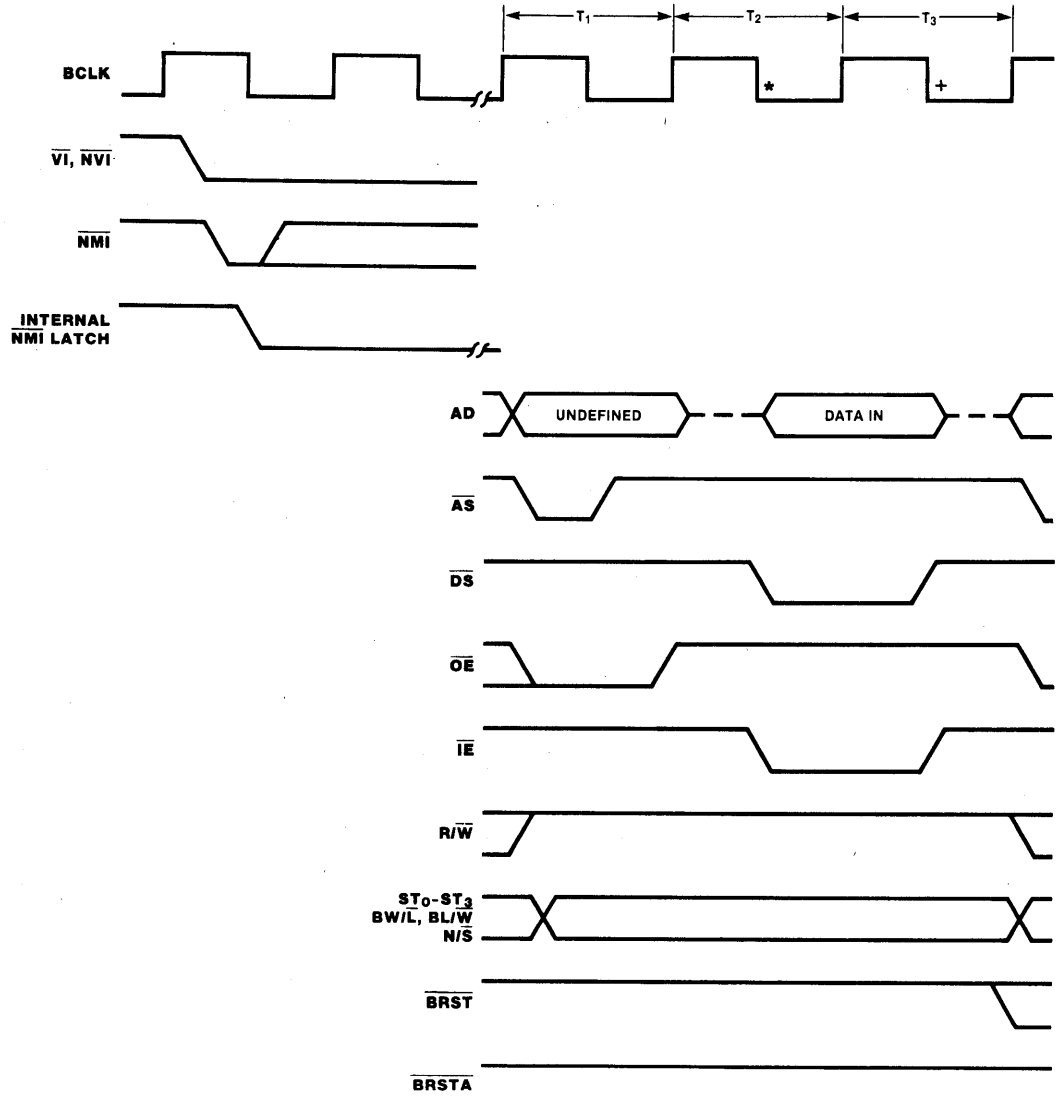
Figure 25. EPU-to-Memory Single Write Timing

INTERRUPT REQUEST AND ACKNOWLEDGE

The CPU recognizes vectored, nonvectored, and nonmaskable interrupt requests. The decreasing order of priority for interrupts is nonmaskable, vectored, and nonvectored. $\overline{NMI}$ is edge sensitive; when $\overline{NMI}$ is asserted, an internal latch is loaded. $\overline{VI}$ and $\overline{NVI}$ are level sensitive.

The CPU samples $\overline{VI}$, $\overline{NVI}$, and the internal $\overline{NMI}$ latch on the rising edge of CLK. The interrupt request signals can be asynchronous to CLK; the CPU synchronizes them internally.

Figure 26 shows timing for an Interrupt Acknowledge transaction, indicated by status 0101, 0110, or 0111. The timing is similar to a single I/O read. Wait states (either programmed for automatic insertion or externally generated) can be inserted before $\overline{DS}$ falls in the middle of T_2 and before $\overline{DS}$ rises in the middle of T_3. Inserting Wait states before $\overline{DS}$ falls allows for delay in the interrupt priority daisy chain. A word of data is transferred. All of the interrupts save the transferred word on the system stack for processing the interrupt. Vectored interrupts use the low-order byte of the word to select a unique PC value from the Program Status Area.



*RSP_0-RSP_1 sampled.
+ RSP_0-RSP_1 and data sampled.

Figure 26. Interrupt Response/Acknowledge Timing

INTERNAL OPERATION AND HALT TRANSACTIONS

Figure 27 shows timing for Internal Operation (status = 0000) and Halt (status = 0011) transactions. Unlike other bus transactions, data is not transferred in these operations.

Nevertheless, the transfer size for these transactions is longword. The minimum duration for the transaction is two bus cycles.

The CPU generates an Internal Operation transaction after a sequence of interlocked memory transactions to free the memory system lock when no other transactions need to be performed. The CPU generates a Halt transaction upon entering Halt State—when the Halt instruction is executed, or when memory indicates Bus Error during a fetch or store of Program Status for exception processing. The CPU leaves Halt state when an interrupt or reset occurs. When the minimum Address Strobe rate option is enabled (controlled by bit MASR in HICR), the CPU maintains a minimum rate for Address Strobes by generating Halt transactions in Halt state or Internal Operation transactions otherwise.

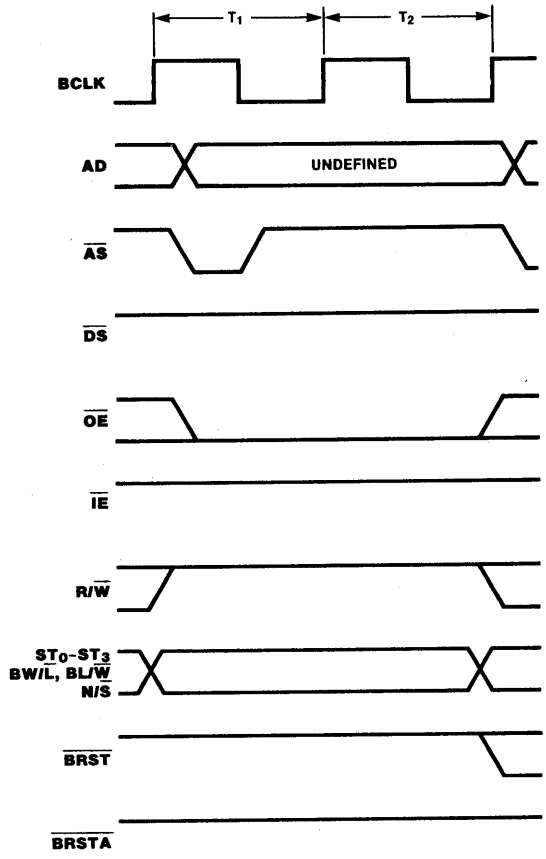


Figure 27. Internal Operation and Halt Timing

BUS RETRY

During any transaction in which data is transferred, the responding device can indicate Bus Retry on RSP_0–RSP_1. When Bus Retry is indicated, the CPU terminates the transaction in progress, negating $\overline{DS}$ and $\overline{BRST}$. If Bus Retry is indicated during a Burst transfer, the Retry transaction begins with the address for the data transfer where Bus Retry is indicated.

BUS ERROR

During any transaction in which data is transferred, the responding device can indicate a bus error exception on RSP_0–RSP_1. When Bus Error is indicated, the CPU terminates the transaction in progress, negating $\overline{DS}$ and $\overline{BRST}$. A bus error exception also causes termination of the instruction in execution. In processing a bus error exception, the CPU saves the Program Status, physical address for the transaction, and a word identifying the status and control signals used for the transaction.

BUS REQUEST AND ACKNOWLEDGE

The CPU supports two types of bus request/acknowledge sequences, local and global. Other bus masters request the local bus from the CPU using a handshake of $\overline{\text{BUSREQ}}$ and $\overline{\text{BUSACK}}$. The CPU requests a global bus from an external arbiter using a handshake of $\overline{\text{GREQ}}$ and $\overline{\text{GACK}}$.

To generate transactions on the local bus, a potential bus master (such as a DMA controller) must gain control of the bus by making a bus request (Figure 28). A local bus request is initiated by asserting $\overline{\text{BUSREQ}}$. Several bus requestors can be wired to the $\overline{\text{BUSREQ}}$ signal; priorities are resolved externally to the CPU, usually by a prioritized daisy chain.

The CPU samples $\overline{\text{BUSREQ}}$ on the rising edge of CLK. $\overline{\text{BUSREQ}}$ can be asynchronous to CLK; the CPU synchronizes it internally. After $\overline{\text{BUSREQ}}$ is asserted, the CPU completes any transaction or sequence of interlocked transactions in progress, including possible retries. Next the CPU responds by asserting $\overline{\text{BUSACK}}$ and placing all other output signals in 3-state. When $\overline{\text{BUSREQ}}$ is negated, the CPU negates $\overline{\text{BUSACK}}$ and begins driving all other output signals.

The CPU can initiate transactions with devices located on a global bus shared with other CPUs. At any time, only one of the CPUs can initiate transactions on the global bus. *Control of the global bus is arbitrated by external* circuitry. Before initiating transactions on the global bus, the CPU requests control of the global bus from the arbiter using the protocol described below. The CPU uses two fields of HICR to distinguish between local and global bus transactions. The GE bit enables use of the global bus. The 4-bit LA field, which is compared with bits 26 to 29 of the physical address, specifies one of sixteen sections of the physical address space to use for local references; other references use the global request protocol.

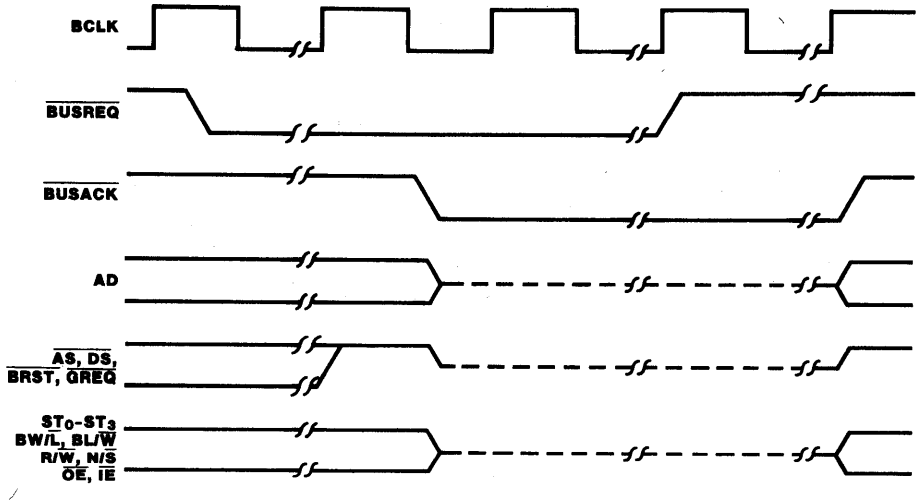


Figure 28. Local Bus Request/Acknowledge Timing

Figure 29 shows timing for the global bus request/acknowledge protocol. Before initiating a transaction on the global bus, the CPU drives the address, ST_0–ST_3, $\overline{BRST}$, R/$\overline{W}$, N/$\overline{S}$, BL/$\overline{W}$, and BW/$\overline{L}$ valid at the beginning of a bus cycle. Then, in the middle of the bus cycle, the CPU asserts $\overline{GREQ}$. When the global bus selected by the address is available to the CPU, the arbiter asserts $\overline{GACK}$. The CPU samples $\overline{GACK}$ on the rising edge of CLK. $\overline{GACK}$ can be asynchronous to CLK; the CPU synchronizes it internally. The CPU performs one or more transactions on the global bus, then negates $\overline{GREQ}$. The arbiter responds by negating $\overline{GACK}$, and the CPU can then initiate more transactions.

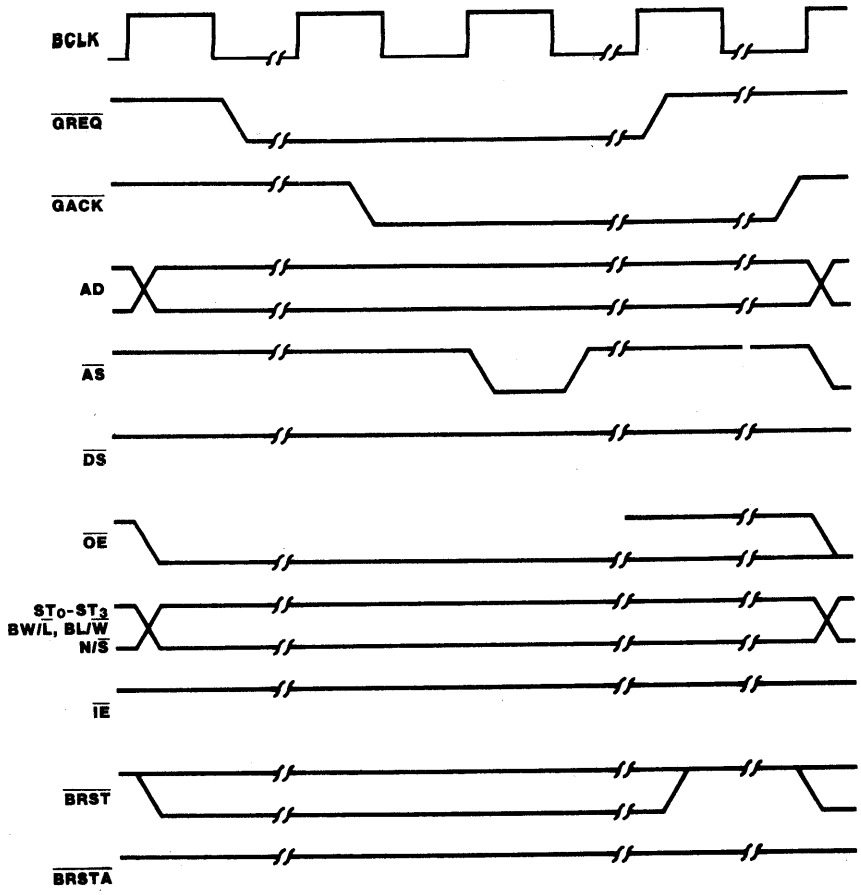
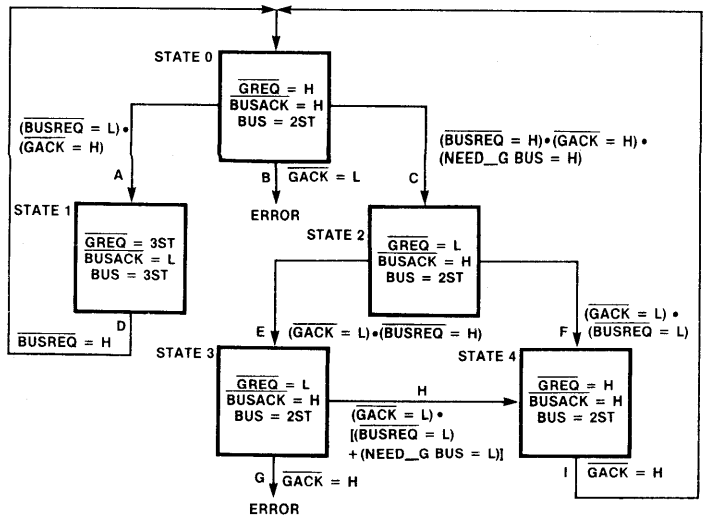


Figure 29. Global Bus Request/Acknowledge Timing

Figure 30 shows a state diagram for the local and global bus request protocols. To prevent deadlock between CPUs referencing each other's local memories, a CPU can be preempted while it is waiting for $\overline{GACK}$ in State 2.

If $\overline{BUSREQ}$ is asserted before $\overline{GACK}$, the CPU relinquishes the global bus without performing any transactions.



NOTES: Interface signals are High (H), Low (L), High or Low (2st), or 3-stated (3st).

Need_G BUS is an active High signal internal to the CPU.

Figure 30. State Diagram for CPU Bus Request Protocol

Transition Legend

A	A local bus request occurs.
B	The global bus arbiter grants control of the global bus when no global bus request is pending. This is an error. The CPU remains in State 0.
C	The CPU requests the global bus in response to the internally generated signal NEED_GBUS.
D	The local bus master relinquishes the bus.
E	The global bus arbiter grants the global bus to the CPU while no local bus request is pending.
F	The global bus arbiter grants the global bus to the CPU while no local bus request is pending. The arbiter has preempted the CPU.
G	The global bus arbiter reclaims the global bus before the CPU relinquishes the global bus. This is an error. The CPU's response to this error is undefined.
H	The CPU relinquishes control of the global bus when it no longer needs the global bus or in response to a local bus request.
I	The global bus arbiter reclaims the global bus.

State Legend

State 0	The CPU controls the local bus and is neither requesting nor controlling the global bus. The CPU can perform transactions on the local bus.
State 1	The CPU has granted the local bus. The CPU cannot perform transactions.
State 2	The CPU controls the local bus and is requesting the global bus. The CPU cannot perform transactions.
State 3	The CPU controls the local and global busses. The CPU can perform transactions on the global bus.
State 4	The CPU controls the local bus and is relinquishing control of the global bus. The CPU cannot perform transactions.

Reset

Figure 31 shows reset timing. After $\overline{RESET}$ is asserted, the CPU responds as follows.

- AD lines are turned to input direction
- $\overline{AS}$, $\overline{BRST}$, $\overline{BUSACK}$, $\overline{DS}$, $\overline{GREQ}$, $\overline{IE}$, and $\overline{OE}$ are negated
- ST_0–ST_3 are driven to 1111
- BW/$\overline{L}$ and BL/$\overline{W}$ are driven Low
- N/$\overline{S}$ and R/$\overline{W}$ are undefined

At power on, $\overline{RESET}$ should be asserted until power has stabilized.

During reset, bits SX, NX, CI, and CD of the SCCL control register are cleared, disabling the address translation and cache mechanisms. Bit GE of HICR is also cleared, disabling the global bus request protocol. At the rising edge of $\overline{RESET}$, the relationship between bus timing, memory data path, and number of automatic Wait states is determined. If RSP_0 is High at the rising edge of $\overline{RESET}$, HICR is initialized to a default configuration of bus clock scale factor 2, word memory data path, and seven automatic Wait states. If RSP_0 is Low at the rising edge of $\overline{RESET}$, HICR is initialized by latching data from the AD lines. $\overline{RESET}$ need not be synchronous with CLK; however, the CPU assumes that the last rising edge of CLK on which $\overline{RESET}$ is asserted corresponds to a rising edge of BCLK. If $\overline{RESET}$ is synchronized with the rising edge of the external bus clock, the internal and external bus clocks will be in phase with respect to CLK. After $\overline{RESET}$ is negated, the CPU fetches FCW from address 2 and PC from address 4.

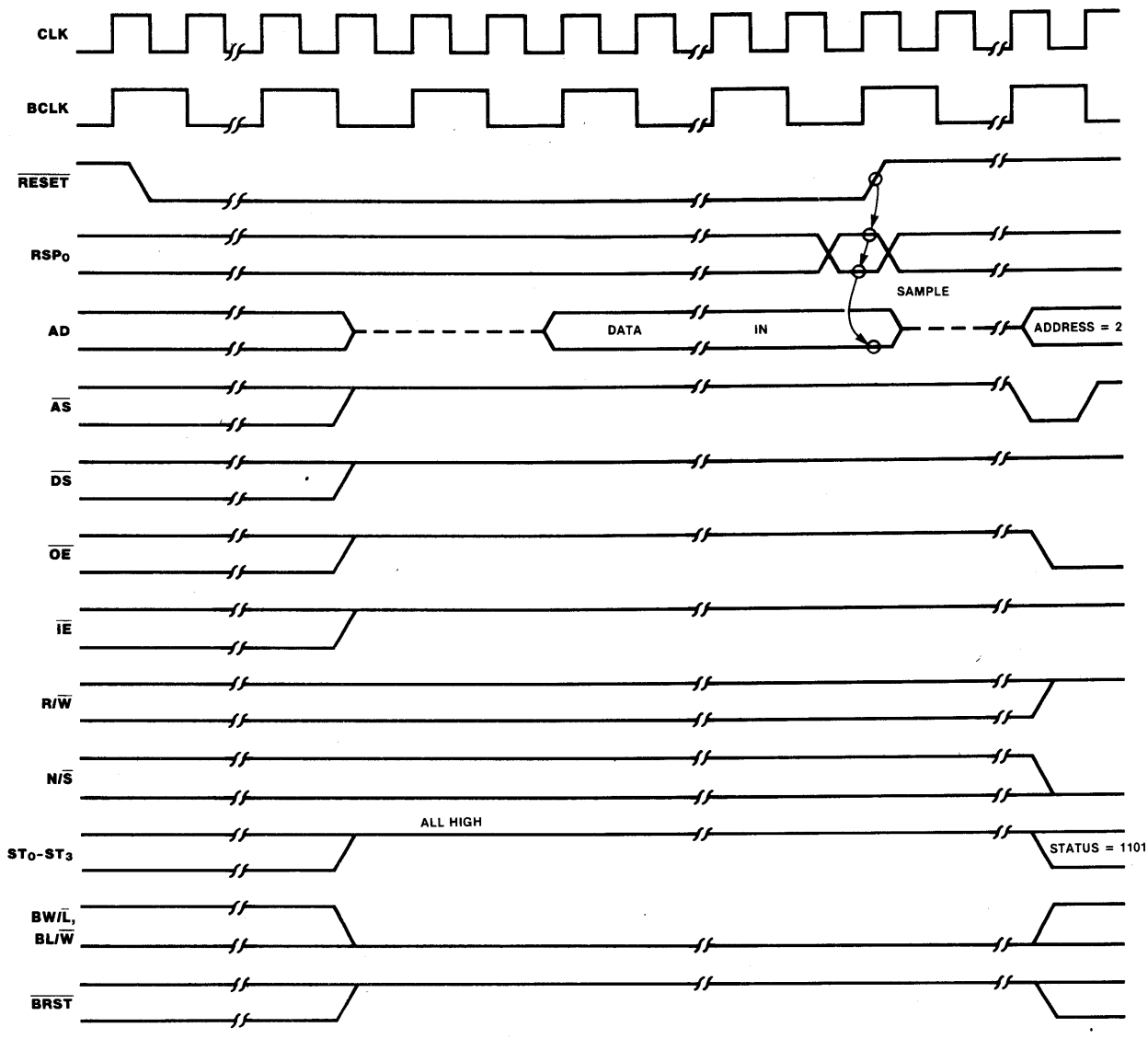


Figure 31. Reset Timing

Additional Information

Zilog

Pioneering the Microworld

Z-BUS®
Component Interconnect

Zilog

Summary

September 1983

Features

- Multiplexed address/data bus shared by memory and I/O transfers.
- 16 or more memory address bits; 16-bit I/O addresses; 8 or 16 data bits.
- Supports polling and vectored or non-vectored interrupts.
- Daisy-chain interrupt structure services interrupts without a separate priority controller.
- Direct addressing of registers within a peripheral facilitates I/O programming.
- Bus signals allow asynchronous CPU and peripheral clocks.
- Daisy-chain bus-request structure supports distributed control of the bus.
- Shared resources can be managed by a general-purpose, distributed resource-request mechanism.

General Description

The Z-BUS is a high-speed parallel shared bus that links components of the Z8000 Family. It provides family members with a common communication interface that supports the following kinds of interactions:

- *Data Transfer.* Data can be moved between bus controllers (such as a CPU) and memories or peripherals.
- *Interrupts.* Interrupts can be generated by peripherals and serviced by CPUs over the bus.
- *Resource Control.* Distributed management of shared resources (including the bus itself) is supported by a daisy-chain priority mechanism.

The heart of the Z-BUS is a set of multiplexed address/data lines and the signals that control these lines. Multiplexing data and addresses onto the same lines makes more efficient use of pins and facilitates expansion of the number of data and address bits. Multiplexing also allows straightforward addressing of a peripheral's internal registers, which greatly simplifies I/O programming.

A daisy-chained priority mechanism resolves interrupt and resource requests, thus allowing distributed control of the bus and eliminating the need for separate priority controllers. The resource-control daisy chain allows wide physical separation of components.

The Z-BUS is asynchronous in the sense that peripherals do not need to be synchronized with the CPU clock. All timing information is provided by Z-BUS signals.

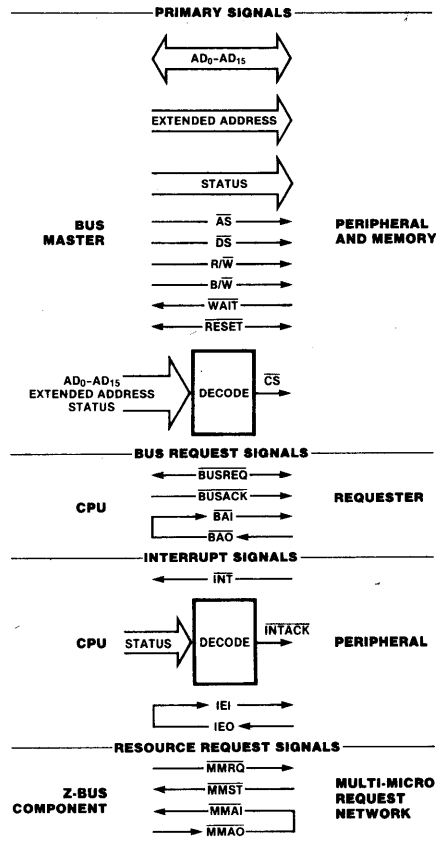


Figure 1. Z-BUS Signals

Z-BUS Components

A Z-BUS component is one that uses Z-BUS signals and protocols, and meets the specified ac and dc characteristics. Most components in the Z8000 Family are Z-BUS components. The four categories of Z-BUS components are as follows:

CPUs. A Z-BUS system contains one CPU, and this CPU has default control of the bus and typically initiates most bus transactions. Besides generating bus transactions, it handles interrupt and bus-control requests. The Z8001 Segmented CPU and Z8002 Non-Segmented CPU are Z-BUS CPUs.

Peripherals. A Z-BUS peripheral is a component capable of responding to I/O transactions and generating interrupt requests. The *Z8036 Counter Input/Output Circuit (Z-CIO), Z8038 FIFO Input/Output, Interface Unit (Z-FIO),* the *Z8030 Serial Communication Controller (Z-SCC),* the *Z8090 Universal Peripheral Controller (Z-UPC),* and the *Z8052 CRT Controller (Z-CRT)* are all Z-BUS peripherals.

Requesters. A Z-BUS requester is any component capable of requesting control of the bus and initiating transactions on the bus. A Z-BUS requester is usually also a peripheral. The *Z8016 DMA Transfer Controller (Z-DTC)* is a Z-BUS requester and a peripheral.

Memories. A Z-BUS memory is one that interfaces directly to the Z-BUS and is capable of fetching and storing data in response to Z-BUS memory transactions. The *Z6132 Quasi-Static RAM* is a Z-BUS memory.

Other Components

The *Z8 Microcomputer*—in its microprocessor configuration—conforms to Z-BUS timing (which allows it to use Z-BUS peripherals and memories), but is missing a wait input and certain status outputs.

The *Z8010 Memory Management Unit (Z-MMU)* is a Z8000 CPU support component that interfaces with part of the Z-BUS on the CPU side and provides demultiplexed addresses on the memory side.

The *Z8060 First-In-First-Out Buffer (Z-FIFO)* is not a Z-BUS component; rather, it is used to expand the buffer depth of the Z-FIO or to interface the I/O ports of the Z-UPC, Z-CIO, or Z-FIO to user equipment.

Z-80 Family components, while not Z-BUS compatible, are easily interfaced to Z-BUS CPUs.

Operation

Two kinds of operations can occur on the Z-BUS: transactions and requests. At any given time, one device (either the CPU or a bus requester) has control of the Z-BUS and is known as the *bus master.* A transaction is initiated by a bus master and is responded to by some other device on the bus. Four kinds of transactions occur in Z-BUS systems:

- ***Memory.*** Transfers 8 or 16 bits of data to or from a memory location.
- ***I/O.*** Transfers 8 or 16 bits of data to or from a peripheral.
- ***Interrupt Acknowledge.*** Acknowledges an interrupt and transfers an identification/status vector from the interrupting peripheral.
- ***Null.*** Does not transfer data. Typically used for refreshing memory.

Only one transaction can proceed on the bus at a time, and it must be initiated by the bus master. A request, however, may be initiated by a component that does not have control of the bus. There are three kinds of requests:

- ***Interrupt.*** Requests the attention of the Z-BUS CPU.
- ***Bus.*** Requests control of the Z-BUS to initiate transactions.
- ***Resource.*** Requests control of a particular resource.

When a request is made, it is answered according to its type: for interrupt requests an interrupt-acknowledge transaction is initiated; for bus and resource requests an acknowledge signal is sent. In all cases a daisy-chain priority mechanism provides arbitration between simultaneous requests.

Signal Lines

The Z-BUS consists of a set of common signal lines that interconnect bus components (Figure 1). The signals on these lines can be grouped into four catagories, depending on how they are used in transactions and requests.

Primary Signals. These signals provide timing, control, and data transfer for Z-BUS transactions.

AD_0–AD_{15}. Address/Data (active High). These multiplexed data and address lines carry I/O addresses, memory addresses, and data during Z-BUS transactions. A Z-BUS may have 8 or 16 bits of data depending on the type of CPU. In the case of an 8-bit Z-BUS, data is transferred on AD_0–AD_7.

Extended Address. (active High). These lines extend AD_0–AD_{15} to support memory addresses greater than 16 bits. The number of lines and the type of address information carried is dependent on the CPU.

Status. (active High). These lines designate the kind of transaction occurring on the bus and certain additional information about the transaction (such as program or data memory access or System versus Normal Mode).

$\overline{AS}$. Address Strobe (active Low). The rising edge of $\overline{AS}$ indicates the beginning of a transaction and that the Address, Status, R/$\overline{W}$, and B/$\overline{W}$ signals are valid.

$\overline{DS}$. Data Strobe (active Low). $\overline{DS}$ provides timing for data movement to or from the bus master.

R/$\overline{W}$. Read/Write (Low = write). This signal determines the direction of data transfer for memory or I/O transactions.

B/$\overline{W}$. Byte/Word (Low = word). This signal indicates whether a byte or word of data is to be transmitted on a 16-bit bus. This signal is not present on an 8-bit bus.

$\overline{WAIT}$. (active Low). A Low on this line indicates that the responding device needs more time to complete a transaction.

$\overline{RESET}$. (active Low). A Low on this line resets the CPU and bus users. Peripherals may be reset by $\overline{RESET}$ or by holding $\overline{AS}$ and $\overline{DS}$ Low simultaneously.

$\overline{CS}$. Chip Select (active Low). Each peripheral or memory component has a $\overline{CS}$ line that is decoded from the address and status lines. A Low on this line indicates that the peripheral or memory component is being addressed by a transaction. The Chip Select information is latched on the rising edge of $\overline{AS}$.

Bus Request Signals. These signals make bus requests and establish which component should obtain control of the bus.

$\overline{BUSREQ}$. Bus Request (active Low). This line is driven by all bus requesters. A Low indicates that a bus requester has or is trying to obtain control of the bus.

$\overline{BUSACK}$. Bus Acknowledge (active Low). A Low on this line indicates that the Z-BUS CPU has relinquished control of the bus in response to a bus request.

$\overline{BAI}$, $\overline{BAO}$. Bus Acknowledge In, Bus Acknowledge Out (active Low). These signals form the bus-request daisy chain.

Z-BUS Connections

Signal	CPU	Requester	Peripheral	Memory
AD_0–AD_{15}	Bidirectional[2] 3-state	Bidirectional[2] 3-state	Bidirectional[1] 3-state	Bidirectional[2] 3-state
Extended Address[8]	Output 3-state	Output 3-state	□	Input
Status	Output 3-state	Output 3-state	Input[10]	□
R/$\overline{W}$	Output 3-state	Output 3-state	Input	Input
B/$\overline{W}$[9]	Output	Output	Input[3]	Input
$\overline{WAIT}$	Input	Input	Output[8] Open Drain	Output[8] Open Drain
$\overline{AS}$	Output 3-state	Output 3-state	Input	Input
$\overline{DS}$	Output 3-state	Output 3-state	Input	Input
$\overline{CS}$[4]	□	□	Input	Input
$\overline{RESET}$	Input	Input[13]	Input[5]	□
$\overline{BUSREQ}$	Input	Bidirectional Open Drain	□	□
$\overline{BUSACK}$	Output	□	□	□
$\overline{BAI}$[7]	□	Input	□	□
$\overline{BAO}$[7]	□	Output	□	□
$\overline{INT}$	Input	□	Output Open Drain	□
$\overline{INTACK}$[6]	□	□	Input[11]	□
IEI[7]	□	□	Input	□
IEO[7]	□	□	Output	□
$\overline{MMRQ}$[12]	Output Open Drain			
$\overline{MMST}$[12]	Input			
$\overline{MMAI}$[7, 12]	Input			
$\overline{MMAO}$[7, 12]	Output			

1. Only AD_0–AD_7, unless peripheral is 16-Bit.
2. For an 8-bit bus, only AD_0–AD_7 are bidirectional.
3. Only for a 16-bit peripheral.
4. Derived signal, one for each peripheral or memory; decoded from status and address lines.
5. Optional—peripherals are typically reset by $\overline{AS}$ and $\overline{DS}$ being Low simultaneously; however, they can have a reset input.
6. Derived signal; decoded from status lines.
7. Daisy-chain lines.
8. Optional signal(s).
9. For 16-bit data bus only.
10. Optional—usually only input on peripherals that are also requesters.
11. May be omitted if peripheral inputs status lines.
12. Optional signal; any component may attach to the resource request lines.
13. Optional signal; a bus requestor may also be reset by $\overline{AS}$ and $\overline{DS}$ going Low and $\overline{BAI}$ being High simultaneously.

□ No Connection

Table 1. Z-BUS Component Connections to Signal Lines. *This table shows how the various Z-BUS components attach to each signal line. When a device is both a bus requester and a peripheral, the attributes in both columns of the table should be combined (e.g., input combined with output and 3-state becomes bidirectional and 3-state.)*

Signal Lines (Continued)

Interrupt Signals. These signals are used for interrupt requests and for determining which interrupting component is to respond to an acknowledge. To support more than one type of interrupt, the lines carrying these signals can be replicated. (The Z8000 CPU supports three types of interrupts: non-maskable, vectored, and non-vectored.)

$\overline{INT}$. Interrupt (active Low). This signal can be driven by any peripheral capable of generating an interrupt. A Low on $\overline{INT}$ indicates that an interrupt request is being made.

$\overline{INTACK}$. Interrupt Acknowledge (active Low). This signal is decoded from the status lines. A Low indicates an interrupt acknowledge transaction is in progress. This signal is latched by the peripheral on the rising edge of $\overline{AS}$.

IEI, IEO. Interrupt Enable In, Interrupt Enable Out (active High). These signals form the interrupt daisy chain.

Resource Request Signals. These signals are used for resource requests. To manage more than one resource, the lines carrying these signals can be replicated. (The Z8000 supports one set of resource request lines.)

$\overline{MMRQ}$. Multi-Micro Request (active Low). This line is driven by any device that can use the shared resource. A Low indicates that a request for the resource has been made or granted.

$\overline{MMST}$. Multi-Micro Status (active Low). This pin allows a device to observe the value of the $\overline{MMRQ}$ line. An input pin other than $\overline{MMRQ}$ facilitates the use of line drivers for $\overline{MMRQ}$.

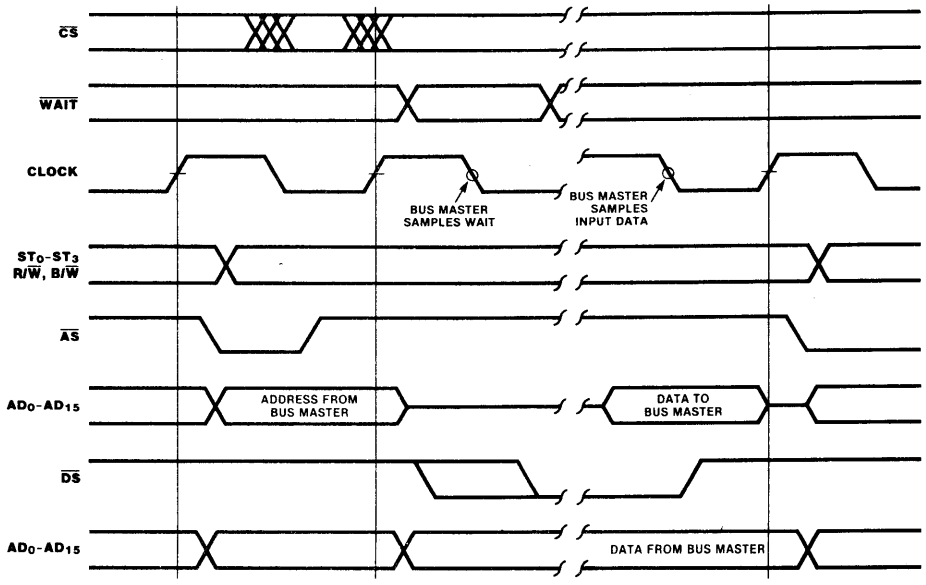
$\overline{MMAI}$, $\overline{MMAO}$. Multi-Micro Acknowledge In, Multi-Micro Acknowledge Out (active Low). These lines form the resource-request daisy chain.

Transactions

All transactions start with Address Strobe being driven Low and then raised High by the bus master (Figure 2).The Status lines are valid on the rising edge of Address Strobe and indicate the type of transactions being initiated. If the transaction requires an address, it must also be valid on the rising edge of Address Strobe.

For all transactions except null transactions (which do nothing beyond this point), data is then transferred to or from the bus master. The bus master uses Data Strobe to time the movement of data. For a read (R/$\overline{W}$ = High), the bus master makes AD_0–AD_{15} inactive before driving Data Strobe Low so that the addressed memory or peripheral can put its data on the bus. The bus master samples this data just before raising Data Strobe High. For a write (R/$\overline{W}$ = Low), the bus master puts the data to be written on AD_0–AD_{15} before forcing Data Strobe Low.

For an 8-bit Z-BUS, data is transferred on AD_0–AD_7. Address bits may remain on AD_8–AD_{15} while $\overline{DS}$ is Low.



Memory Transactions

For a memory transaction, the Status lines distinguish among various address spaces, such as program and data or system and normal, as well as indicating the type of transaction. The memory address is put on AD_0–AD_{15} and on the extended address lines.

For a Z-BUS with 16-bit data, the memory is organized as two banks of eight bits each (Figure 3). One bank contains all the upper bytes of all the addressable 16-bit words. The other bank contains all the lower bytes. When a single byte is written (R/$\overline{W}$ = Low, B/$\overline{W}$ = High), only the bank indicated by address bit A_0 is enabled for writing.

For a Z-BUS with 8-bit data, the memory is organized as one bank which contains all bytes. This bank always inputs and outputs its data on AD_0–AD_7.

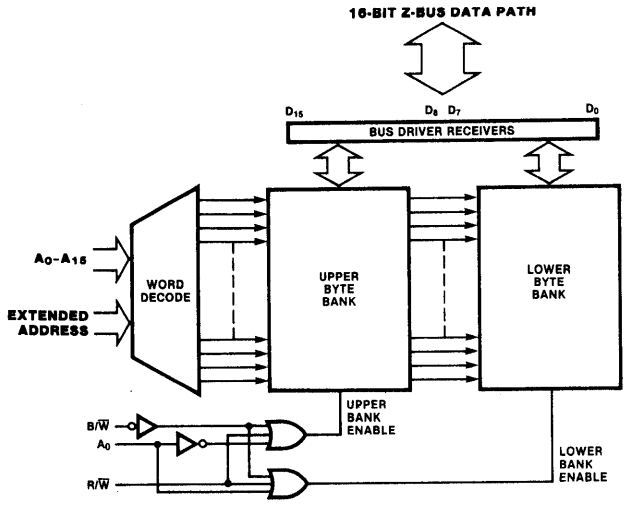


Figure 3. Byte/Word Memory Organization

I/O Transactions

I/O transactions are similar to memory transactions with two important differences. The first is that I/O transactions take an extra clock cycle to allow for slow peripheral operation. The second is that byte data (indicated by B/$\overline{W}$ High on a 16-bit bus) is always transmitted on AD_0–AD_7, regardless of the I/O address. (AD_8–AD_{15} contain arbitrary data in this case.) For an I/O transaction, the address indicates a peripheral and a particular register or function within that peripheral.

Null Transactions

The two kinds of null transactions are distinguished by the Status lines: internal operation and memory refresh. Both transactions look like a memory read transaction except that Data Strobe remains High and no data is transferred.

For an internal operation transaction, the Address lines contain arbitrary data when Address Strobe goes High. This transaction is initiated to maintain a minimum transaction rate when a bus master is doing a long internal operation (to support memories which generate refresh cycles from Address Strobe).

For a memory refresh transaction, the Address lines contain a refresh address when Address Strobe goes High. This transaction is used to refresh a row of a dynamic memory.

Any memory or I/O transaction can be suppressed (effectively turning it into a null transaction) by keeping Data Strobe High throughout the transaction.

Interrupts

A complete interrupt cycle consists of an interrupt request followed by an interrupt-acknowledge transaction. The request, which consists of $\overline{INT}$ pulled Low by a peripheral, notifies the CPU that an interrupt is pending. The interrupt-acknowledge transaction, which is initiated by the CPU as a result of the request, performs two functions: it selects the peripheral whose interrupt is to be acknowledged, and it obtains a vector that identifies the selected device and cause of interrupt.

A peripheral can have one or more sources of interrupt. Each interrupt source has three bits that control how it generates interrupts. These bits are an Interrupt Pending bit (IP), and Interrupt Enable bit (IE), and an Interrupt Under Service bit (IUS).

A peripheral may also have one or more vectors for identifying the source of an interrupt during an interrupt-acknowledge transaction. Each interrupt source is associated with one interrupt vector and each interrupt vector can have one or more interrupt sources associated with it. Each vector has a Vector Includes Status bit (VIS) controlling its use.

Finally, each peripheral has three bits for controlling interrupt behavior for the whole device. These are a Master Interrupt Enable bit (MIE), a Disable Lower Chain bit (DLC), and a No Vector bit (NV).

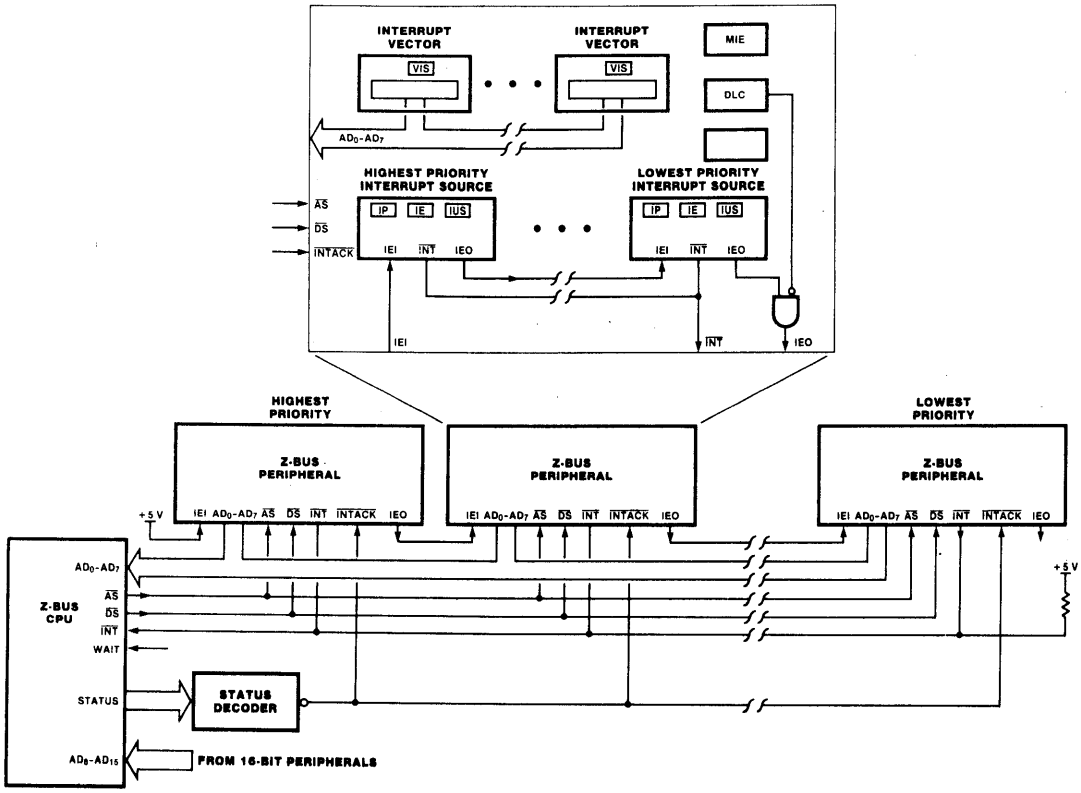


Figure 4. Interrupt Connections

Interrupts (Continued)

Peripherals are connected together via an interrupt daisy chain formed with their IEI and IEO pins (Figure 4). The interrupt sources within a device are similarly connected into this chain with the overall effect being a daisy chain connecting the interrupt sources. The daisy chain has two functions: during an interrupt-acknowledge transaction, it determines which interrupt source is being acknowledged; at all other times it determines which interrupt sources can initiate an interrupt request.

Figure 5 is a state diagram for interrupt processing for an interrupt source (assuming its IE bit is 1). An interrupt source with an interrupt pending (IP = 1) makes an interrupt request (by pulling $\overline{INT}$ Low) if, and only if, it is enabled (IE = 1, MIE = 1), it does not have an interrupt under service (IUS = 0), no higher priority interrupt is being serviced (IEI = High), and no interrupt-acknowledge transaction is in progress (as indicated by $\overline{INTACK}$ at the last rising edge of $\overline{AS}$). IEO is not pulled down by the interrupt source at this time; IEO continues to follow IEI until an interrupt-acknowledge transaction occurs.

Some time after $\overline{INT}$ has been pulled Low, the CPU initiates an interrupt-acknowledge transaction (indicated by $\overline{INTACK}$ Low). Between the rising edge of $\overline{AS}$ and the falling edge of $\overline{DS}$, the IEI/IEO daisy chain settles. Any interrupt source with an interrupt pending (IP = 1, IE = 1, MIE = 1) or under service (IUS = 1) holds its IEO line Low; all other interrupt sources make IEO follow IEI. When $\overline{DS}$ falls, only the highest priority interrupt source with a pending interrupt (IP = 1) has its IEI input High, its IE bit set to 1, and its IUS bit set to 0. This is the interrupt source being acknowledged, and at this point it sets its IUS bit to 1, and, if the peripheral's NV bit is 0, identifies itself by placing the vector on AD_0-AD_7. If the NV bit is 1, then the peripheral's $AD_0 - AD_7$ pins remain floating, thus allowing external circuitry to supply the vector. (All interrupts, including the Z8000's nonvectored interrupt, need a vector for identifying the source of an interrupt.) If the vector's VIS bit is 1, the vector will also contain status information further identifying the source of the interrupt. If the VIS bit is 0, the vector held in the peripheral will be output without modification.

While an interrupt source has an interrupt under service (IUS = 1), it prevents all lower priority interrupt sources from requesting interrupts by forcing IEO Low. When interrupt servicing is complete, the CPU must reset the IUS bit and, in most cases, the IP bit (by means of an I/O transaction).

A peripheral's Master Interrupt Enable bit (MIE) and Disable Lower Chain bit (DLC) can modify the behavior of the peripheral's interrupt sources in the following way: if the MIE bit is 0, the effect is as if every Interrupt Enable bit (IE) in the peripheral were 0; thus all interrupts from the peripheral are disabled. If the DLC bit is 1, the effect is to force the peripheral's IEO output Low, thus disabling all lower priority devices from initiating interrupt requests.

Polling can be done by disabling interrupts (using MIE and DLC) and by reading peripherals to detect pending interrupts. Each Z-BUS peripheral has a single directly addressable register that can be read to determine if there is an interrupt pending in the device and, if so, what interrupt source it is from.

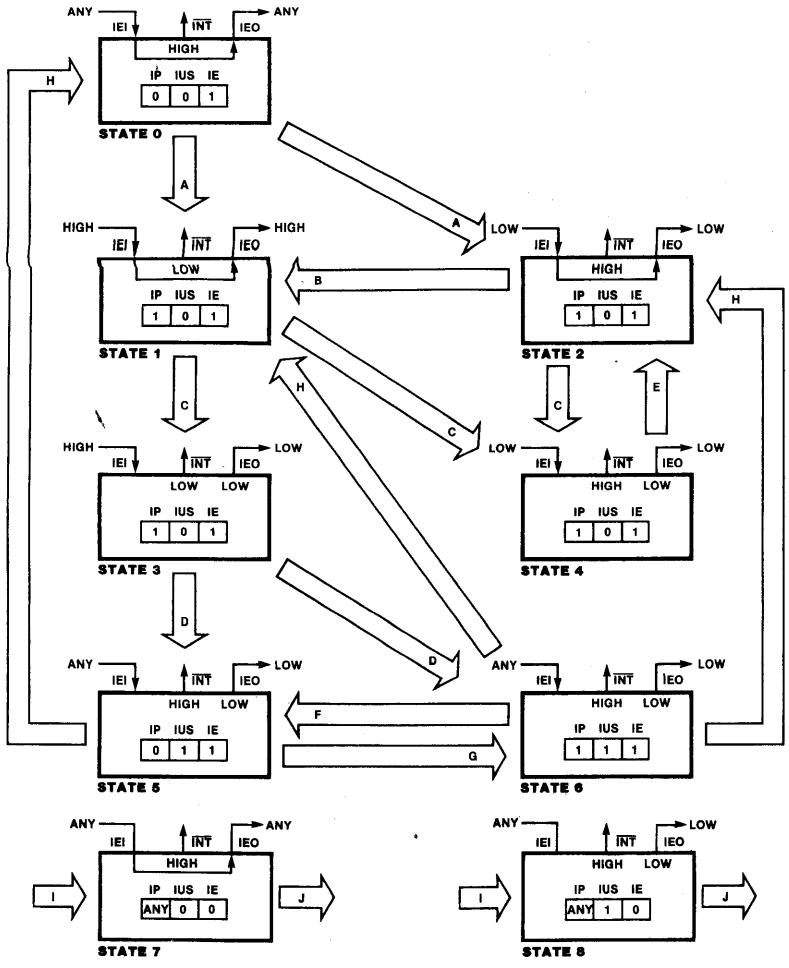


Figure 5. State Diagram for an Interrupt Source

Transition Legend

- A: The peripheral detects an interrupt condition and sets Interrupt Pending.
- B: All higher priority peripherals finish interrupt service, thus allowing IEI to go High.
- C: An interrupt-acknowledge transaction starts, and the IEI/IEO daisy chain settles.
- D: The interrupt-acknowledge transaction terminates with the peripheral selected. Interrupt Under Service (IUS) is set to 1, and Interrupt Pending (IP) may or may not be reset.
- E: The interrupt-acknowledge transaction terminates with a higher priority device having been selected.
- F: The Interrupt Pending bit in the peripheral is reset by an I/O operation.
- G: A new interrupt condition is detected by the peripheral, causing IP to be set again.
- H: Interrupt service is terminated for the peripheral by resetting IUS.
- I[3]: IE is reset to zero, causing interrupts to be disabled.
- J[4]: IE is set to one, re-enabling interrupts.

State Legend

- 0: No interrupts are pending or under service for this peripheral.
- 1: An interrupt is pending, and an interrupt request has been made by pulling $\overline{INT}$ Low.
- 2: An interrupt is pending, but no interrupt request has been made because a higher priority peripheral has an interrupt under service, and this has forced IEI Low.
- 3: An interrupt-acknowledge sequence is in progress, and no higher priorify peripheral has a pending interrupt.
- 4: An interrupt-acknowledge sequence is in progress, but a higher priority peripheral has a pending interrupt, forcing IEI Low.
- 5: The peripheral has an interrupt under service. Service may be temporarily suspended (indicated by IEI going Low) if a higher priority device generates an interrupt.
- 6: This is the same as State 5 except that an interrupt is also pending in the peripheral.
- 7: Interrupts are disabled from this source because IE = 0.
- 8: Interrupts are disabled from this source and lower priority sources because IE = 0 and IUS = 1.

1. This diagram assumes MIE = 1. The effect of MIE = 0 is the same as that of setting IE = 0.
2. The DLC bit does not affect the states of individual interrupt sources. Its only effect is on the IEO output of a whole peripheral.
3. Transition I to state 6 or 7 can occur from any state except 3 or 4 (which only occur during interrupt acknowledge).
4. Transition J from state 6 or 7 can be to any state except 3 or 4, depending on the value of IEI, IP, and IUS.

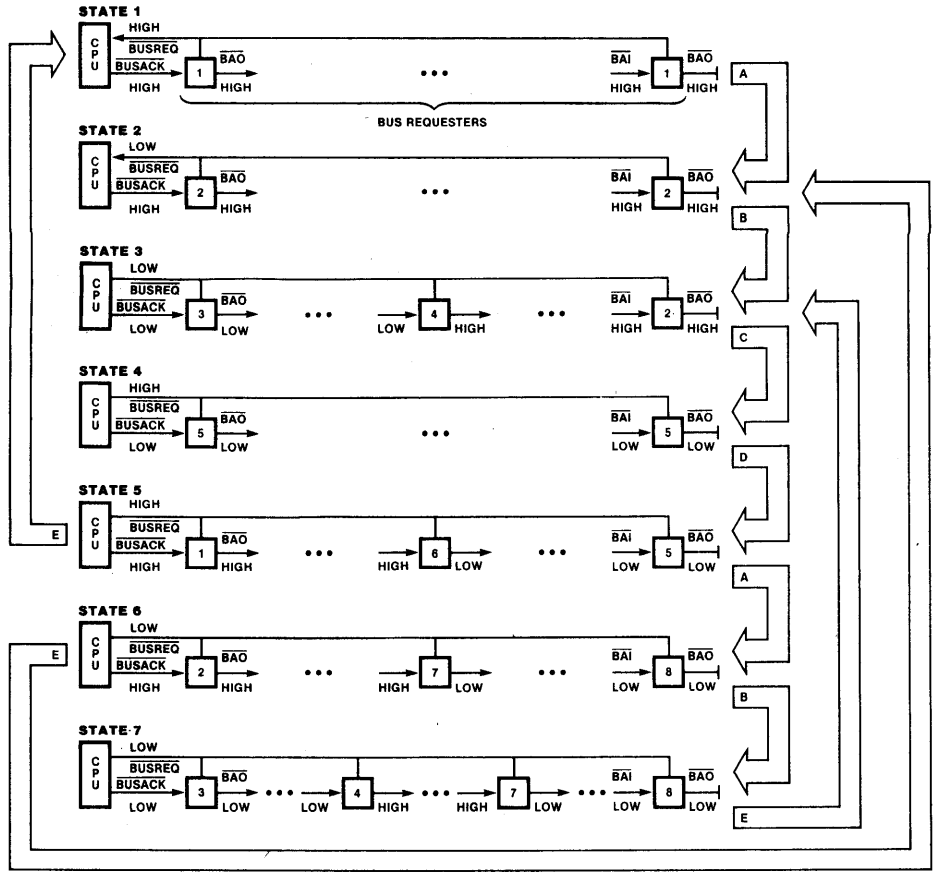


Figure 6. Bus Request Mechanism States

Bus Requester Legend

1. Requester does not want bus and is not pulling $\overline{\text{BUSREQ}}$ Low.
2. Requester may or may not want bus; it is pulling $\overline{\text{BUSREQ}}$ Low in either case.
3. Requester is not pulling $\overline{\text{BUSREQ}}$ Low; it it wants control of the bus, it must wait for $\overline{\text{BUSREQ}}$ and $\overline{\text{BAI}}$ to rise before requesting the bus.
4. Requester is either using the bus or propagating the Low on its $\overline{\text{BAI}}$ input. It will stop driving $\overline{\text{BUSREQ}}$ when its $\overline{\text{BAO}}$ output goes Low. If it wants to use the bus, but did not want to at the time $\overline{\text{BUSREQ}}$ and $\overline{\text{BAI}}$ were last High or $\overline{\text{BUSREQ}}$ went from Low to High, then it must wait for $\overline{\text{BUSREQ}}$ and $\overline{\text{BAI}}$ to rise before requesting and using the bus.
5. Requester is not pulling $\overline{\text{BUSREQ}}$ Low. If it wants to use the bus, it must wait for its $\overline{\text{BAI}}$ to become High before requesting the bus.
6. Requester is propagating the High on its $\overline{\text{BAI}}$ input. If it wants the bus it will pull $\overline{\text{BUSREQ}}$ Low.
7. Requester is propagating the High on its $\overline{\text{BAI}}$ input.
8. Requester is not pulling $\overline{\text{BUSREQ}}$ Low. If it wanted the bus at the time $\overline{\text{BUSREQ}}$ went from Low to High, it may request the bus when its $\overline{\text{BAI}}$ input rises; otherwise if it wants the bus, it must wait for $\overline{\text{BUSREQ}}$ to rise.

Bus State Legend

1. The CPU owns the bus and no one is requesting it.
2. A bus requester has requested the bus by pulling $\overline{\text{BUSREQ}}$ Low, but the CPU has nor responded.
3. A Low from the CPU's $\overline{\text{BUSACK}}$ is propagating down the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain. Bus requesters are using the bus.
4. The Low from $\overline{\text{BUSACK}}$ has propagated to the end of the daisy chain causing all bus requesters to release $\overline{\text{BUSREQ}}$, which floats High. The CPU has not yet acknowledged return of the bus.
5. The CPU acknowledges the High on $\overline{\text{BUSREQ}}$ with a High on $\overline{\text{BUSACK}}$, which has propagated down the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain.
6. Some device whose $\overline{\text{BAI}}$ input is High requests the bus by pulling $\overline{\text{BUSREQ}}$ Low. The CPU has not yet responded with a Low on $\overline{\text{BUSACK}}$.
7. The CPU has responded to a Low on $\overline{\text{BUSREQ}}$ with a Low on $\overline{\text{BUSACK}}$. The previous High state on $\overline{\text{BUSACK}}$ is still propagating down the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain.

Interrupts (Continued)

Transition Legend

A A bus requester requests the bus by pulling down on $\overline{\text{BUSREQ}}$.

B The CPU reponds to $\overline{\text{BUSREQ}}$ by pulling down $\overline{\text{BUSACK}}$.

C The Low from $\overline{\text{BUSACK}}$ propagates to the end of the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain, causing all the bus requesters to let $\overline{\text{BUSREQ}}$ rise.

D The CPU responds to $\overline{\text{BUSREQ}}$ High by driving $\overline{\text{BUSACK}}$ High.

E The High from $\overline{\text{BUSREQ}}$ propagates to the end of the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain.

Bus Requests

Figure 7 shows how the bus request lines connect bus requesters and the CPU on a Z-BUS. Figure 8 shows the states of the bus request mechanism as the Z-BUS is acquired, used, and released.

To generate transactions on the bus, a bus requester must gain control of the bus by making a bus request. This is done by pulling down $\overline{\text{BUSREQ}}$. A bus request can be made in either of two cases:

- $\overline{\text{BUSREQ}}$ is initially High and $\overline{\text{BAI}}$ is High, indicating that the bus is controlled by the CPU and no other requester is requesting the bus.
- $\overline{\text{BAI}}$ is High and the requester had wanted to request the bus at the time of the last Low-to-High transition of $\overline{\text{BUSREQ}}$. This insures that a module will not be locked out indefinitely by a higher priority bus requester.

After $\overline{\text{BUSREQ}}$ is pulled Low, the Z-BUS CPU relinquishes the bus and indicates this condition by making $\overline{\text{BUSACK}}$ Low. The Low on $\overline{\text{BUSACK}}$ is propagated through the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain (Figure 7). $\overline{\text{BAI}}$ follows $\overline{\text{BAO}}$ for components not requesting the bus, and any component requesting the bus holds its $\overline{\text{BAO}}$ High, thereby locking out all lower priority requesters. A bus requester gains control of the bus when its $\overline{\text{BAI}}$ input goes Low. When it is ready to relinquish the bus, it stops pulling $\overline{\text{BUSREQ}}$ Low and allows $\overline{\text{BAO}}$ to follow $\overline{\text{BAI}}$. This permits lower priority devices that made simultaneous requests to gain control of the bus. When all simultaneously requesting devices have relinquished the bus, and the Low on $\overline{\text{BAI}}/\overline{\text{BAO}}$ has propagated to the lowest priority requester, $\overline{\text{BUSREQ}}$ goes High, returning control of the bus to the CPU.

The CPU responds to the High on $\overline{\text{BUSREQ}}$ by driving BUSACK High. The High on $\overline{\text{BUSACK}}$ is propagated down the $\overline{\text{BAI}}/\overline{\text{BAO}}$ daisy chain, thus allowing bus requesters to make new bus requests. Because high priority bus requesters can pull $\overline{\text{BUSREQ}}$ Low before low priority devices have a High on $\overline{\text{BAI}}$, a way is needed for low priority devices to request the bus when $\overline{\text{BUSREQ}}$ is Low. That is provided by the rule that a requester may request the bus if $\overline{\text{BAI}}$ is High and it had wanted the bus at the time the last Low-to-High transition on $\overline{\text{BUSREQ}}$.

As soon as $\overline{\text{BUSREQ}}$ is pulled Low by any requester, each of the other requesters on the bus drives $\overline{\text{BUSREQ}}$ Low and continues to do so until it drives its $\overline{\text{BAO}}$ output Low. This provides a handshake between the CPU and the bus requesters by ensuring that $\overline{\text{BUSREQ}}$ will not go High until the CPU's acknowledgement of $\overline{\text{BUSACK}}$ has reached every requester. Bus requesters can therefore run asynchronously to the CPU. This rule also allows the bidirectional $\overline{\text{BUSREQ}}$ line to be buffered using the logic shown in Figure 8. This logic is similar to the logic inside a bus requester that keeps $\overline{\text{BUSREQ}}$ Low when it has initially been pulled Low by a different requester.

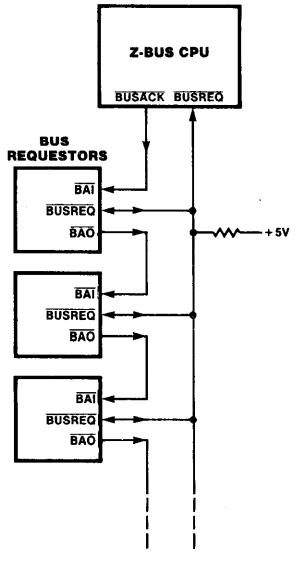


Figure 7. Bus Request Connections

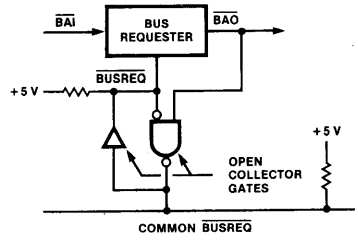


Figure 8. Bus Request Line Buffering

Resource Requests

Resource requests are used to obtain control of a resource that is shared between several users. The resource can be a common bus, a common memory or any other resource. The requestor can be any component capable of implementing the request protocol.

Unlike the Z-BUS itself, no component has control of a general resource by default; every device must acquire the resource before using it. All devices sharing the general resource drive the $\overline{MMRQ}$ line (Figure 9). When Low, the $\overline{MMRQ}$ line indicates that the resource is being acquired or used by some device. The $\overline{MMST}$ pin allows each device to observe the state of the $\overline{MMRQ}$ line.

When $\overline{MMRQ}$ is High, a device may initiate a resource request by pulling $\overline{MMRQ}$ Low (Figure 10). The resulting Low on $\overline{MMRQ}$ is propagated through the $\overline{MMAI}$/$\overline{MMAO}$ daisy chain. If a device is not requesting the resource, its $\overline{MMAO}$ output follows its $\overline{MMAI}$ input. Any device making a resource request forces its $\overline{MMAO}$ output High to deny use of the resource to lower priority devices.

A device gains control of the resource if its $\overline{MMAI}$ input is Low (and its $\overline{MMAO}$ output is High) after a sufficient delay to let the daisy chain settle. If the device does not obtain the resource after this short delay, it must stop pulling $\overline{MMRQ}$ Low and make another request at some later time when $\overline{MMRQ}$ is again High. When a device that has gained control of a resource is finished, it releases the resource by allowing $\overline{MMRQ}$ to go High.

The four unidirectional lines of the resource request chain allow the use of line drivers, thus facilitating connection of components separated by some distance. In the case of the Z8000 CPU, the four resource request lines may be mapped into the CPU $\overline{MI}$ and $\overline{MO}$ pins using the logic shown in Figure 11. With this configuration, the Multi-Micro Request Instruction (MREQ) performs a resource request.

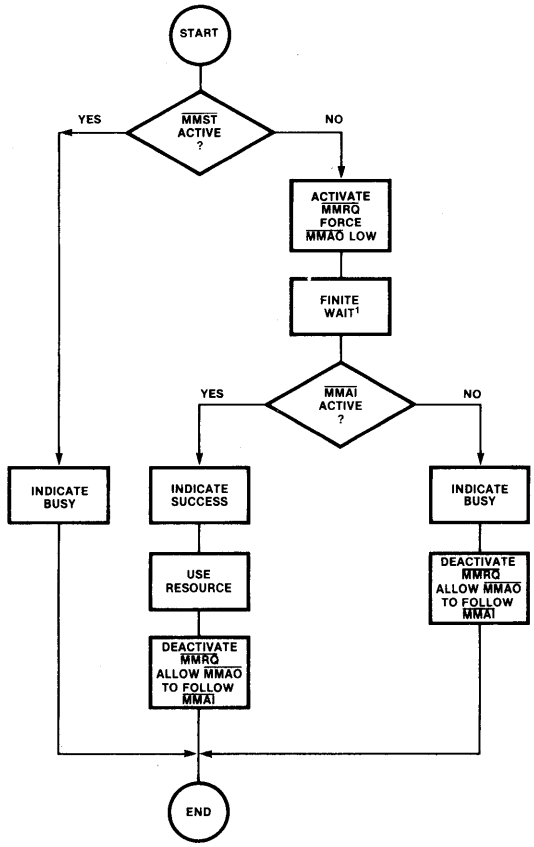


Figure 10. Resource Request Protocol

1. For any resource requested, this wait time must be less than the minimum wait time plus resource usage time of all other requesters.

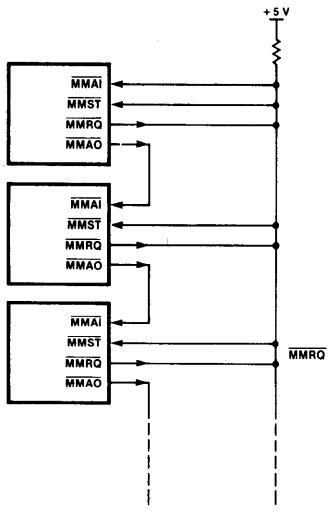


Figure 9. Resource Request Connections

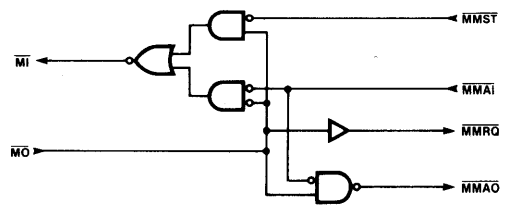
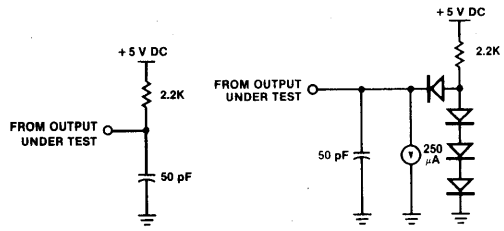


Figure 11. Bus Request Logic for Z8000

Test Conditions

The timing characteristics given in this document reference 2.0 V as High and 0.8 V as Low. The following test load circuit is assumed. The effect of larger capacitive loadings can be calculated by delaying output signal transitions by 10 ns for each additional 50 pF of load up to a maximum 200 pF.



Open-Drain Test Load **Standard Test Load**

DC Characteristics

The following table states the dc characteristics for the input and output pins of Z-BUS components. All voltages are relative to ground.

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	$V_{CC}+0.3$	V	
$V_{IHRESET}$	Input High Voltage on $\overline{RESET}$ pin	2.4	V_{CC} to 0.3	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 250µA
I_{IL}	Input Leakage Current	-10	+10	µA	V_{IN} = 0.4 to 2.4 V
I_{OL}	3-State Output Leakage Current in Float	-10	+10	µA	V_{OUT} = 0.4 to 2.4 V

Capacitance

The following table gives maximum pin capacitance for Z-BUS components. Capacitance is specified at a frequency of 1 MHz over the temperature range of the component. Unused pins are returned to ground.

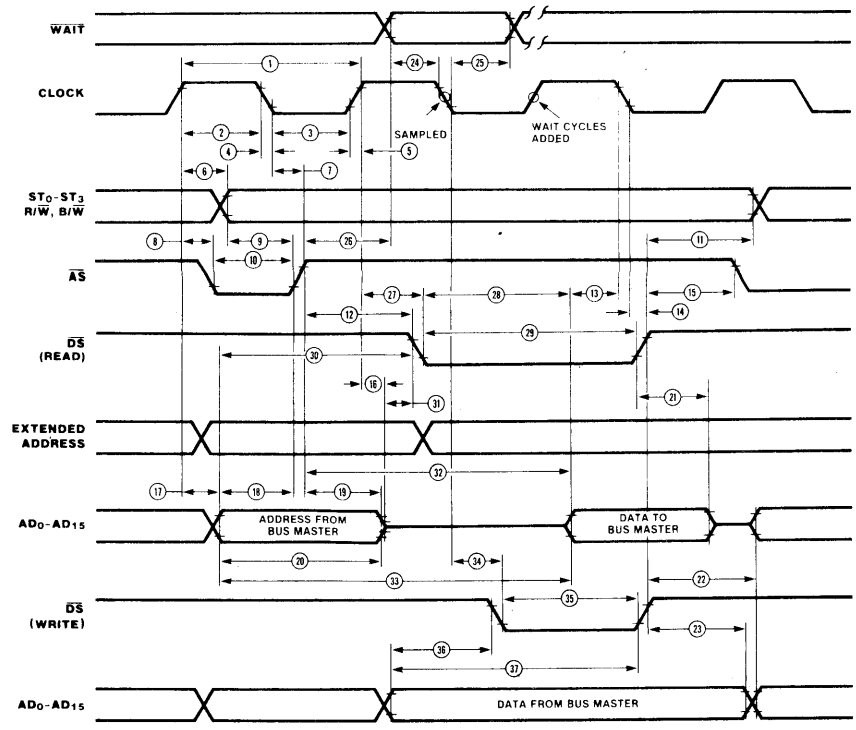
Symbol	Parameter	Max (pF)
C_{IN}	Input Capacitance	10
C_{OUT}	Output Capacitance	15
$C_{I/O}$	Bidirectional Capacitance	15

Timing Diagrams

The following diagrams and tables give the timing for each kind of transaction (except null transactions). Timings are given separately for bus masters and for peripherals and memories and are intended to give the minimum timing requirements which a Z-BUS component must meet. An individual component will have more detailed and sometimes more stringent timing specifications. The differences between bus master timing and peripheral and memory timing allow for buffer and decoding circuit delays and for signal skew. The timing given for memories is a constraint on bus-compatible memories (like the Z6132 Quasi-Static RAM) and is not intended to constrain memory subsystems constructed from conventional components.

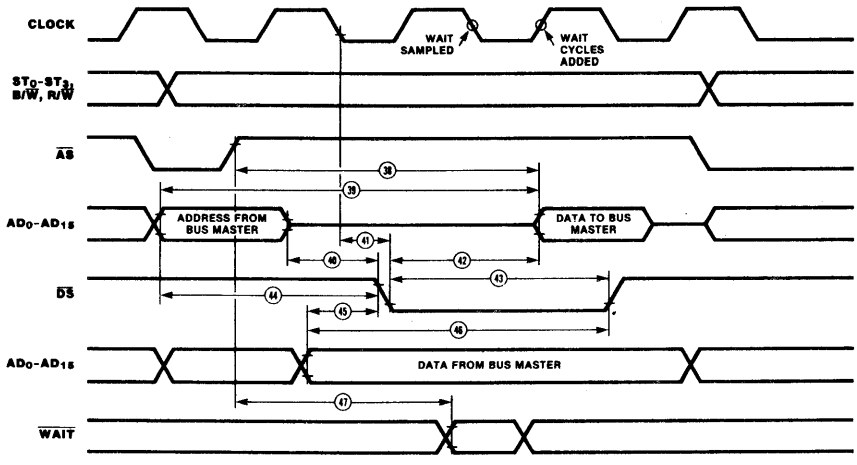
Besides these timings, there is a requirement that at least 128 transactions be initiated in any 2 ms period. This accommodates memories that generate refresh cycles from Address Strobe.

Bus Master Timing

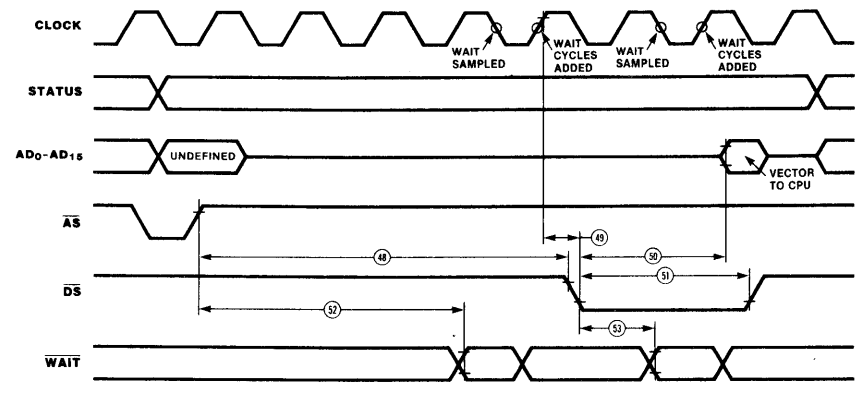


Parameters 1-25 are common to all transactions.

I/O Transaction Timing



Interrupt Acknowledge Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
		All Transactions					
1	TpC	Clock Period	250	2000	165	2000	
2	TwCh	Clock High Width	105		70		
3	TwCl	Clock Low Width	105		70		
4	TfC	Clock Fall Time		20		10	
5	TrC	Clock Rise Time		20		15	
6	TdC(S)	Clock ↑ to Status Valid Delay		110		85	
7	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		80	
8	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		60	
9	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	50		30		
10	TwAS	$\overline{AS}$ Low Width	80		55		
11	TdDS(S)	$\overline{DS}$ ↑ to Status Not Valid Delay	75		55		
12	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	80	2095	55		3
13	TsDR(C)	Read Data to Clock ↓ Setup Time	30		20		
14	TdC(DS)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65	
15	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		35		
16	TdC(Az)	Clock ↑ to Address Float Delay		65		55	
17	TdC(A)	Clock ↑ to Address Valid Delay		100		75	
18	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	50		35		1
19	TdAS(A)	$\overline{AS}$ ↑ to Address Not Valid Delay	70		45		1
20	TwA	Address Valid Width	150		85		
21	ThDR(DS)	Read Data to $\overline{DS}$ ↑ Hold Time	0		0		
22	TdDS(A)	$\overline{DS}$ ↑ to Address Active Delay	80		45		
23	TdDS(DW)	$\overline{DS}$ ↑ to Write Data Not Valid Delay	50		45		
24	TsW(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		30		2,4
25	ThW(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		10		2,4
		Memory Transactions					
26	TdAS(W)	$\overline{AS}$ ↑ to $\overline{WAIT}$ Required Valid		90		45	
27	TdC(DSR)	Clock ↓ to $\overline{DS}$ (Read) ↓ Delay		120		85	
28	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Required Valid		200		130	
29	TwDSR	$\overline{DS}$ (Read) Low Width		250	185		
30	TdA(DS)	Address Valid to $\overline{DS}$ ↓ Delay	180		110		
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0		
32	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		360		220	
33	TdA(DR)	Address Valid to Read Data Required Valid		410		305	
34	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		80	
35	TwDSW	$\overline{DS}$ (Write) Low Width	160		110		
36	TdDW(DSWf)	Write Data Valid to $\overline{DS}$ (Write) ↓ Delay	50		35		
37	TdDW(DSWr)	Write Data Valid to $\overline{DS}$ (Write) ↑ Delay	230		195		
		I/O Transactions					
38	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Required Valid		610		385	
39	TdA(DR)	Address Valid to Read Data Required Valid		660		470	
40	TdAz(DSI)	Address Float to $\overline{DS}$ (I/O) ↓	0		0		
41	TdC(DSI)	Clock ↓ to $\overline{DS}$ (I/O) ↓		120		90	
42	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Required Valid		330		210	
43	TwDSI	$\overline{DS}$ (I/O) Low Width	400		255		
44	TdA(DSI)	Address Valid to $\overline{DS}$ (I/O) ↓ Delay	180		110		
45	TdDW(DSIf)	Write Data to $\overline{DS}$ (I/O) ↓ Delay	50		35		
46	TdDW(DSIr)	Write Data to $\overline{DS}$ (I/O) ↑ Delay	480		320		
47	TdAS(W)	$\overline{AS}$ to $\overline{WAIT}$ Required Valid		340		210	
		Interrupt-Acknowledge Transactions					
48	TdAS(DSA)	$\overline{AS}$ ↑ to DS (Acknowledge) ↓ Delay	960		690		
49	TdC(DSA)	Clock ↑ to DS (Acknowledge) ↓ Delay		120		85	
50	TdDSA(DR)	DS (Acknowledge) ↓ to Read Data Required Valid		455		295	
51	TwDSA	DS (Acknowledge) Low Width	485		315		
52	TdAS(W)	$\overline{AS}$ ↑ to Wait Required Valid		840		540	
53	TdDSA(W)	DS (Acknowledge) ↓ to Wait Required Valid		185		120	

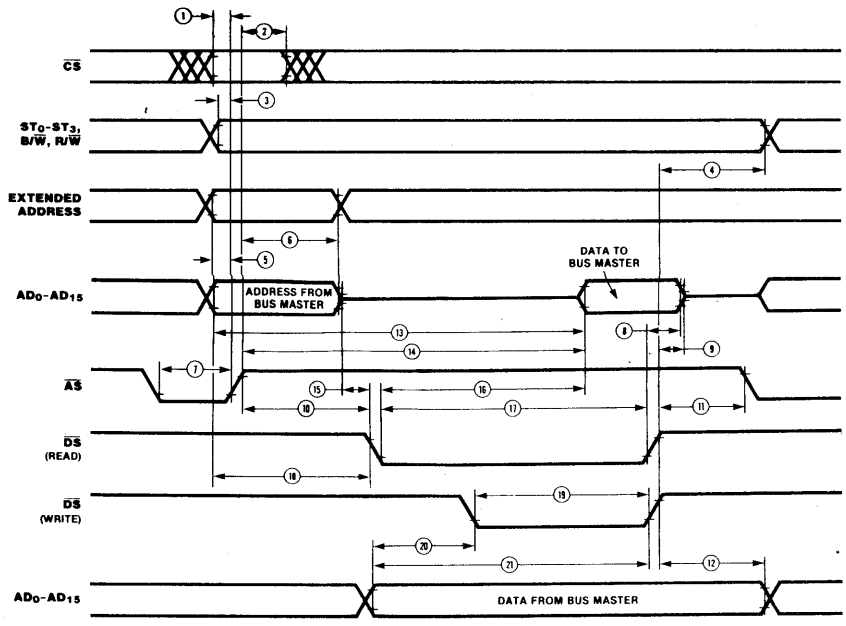
NOTES:
1. Timing for extended addresses is CPU dependent; however, extended addresses must be valid at least as soon as addresses are valid on AD_0-AD_{15} and must remain valid at least as long as addresses are valid on AD_0-AD_{15}.
2. The exact clock cycle that wait is sampled on depends on the type of transaction; however, wait always has the given setup and hold times to the clock.
3. The maximum value for TdAS(DS) does not apply to Interrupt-Acknowledge Transactions.
4. The setup and hold times for $\overline{WAIT}$ to the clock must be met. If $\overline{WAIT}$ is generated asychronously to the clock, it must be synchronized before input to a bus master.

* Timings are preliminary and subject to change.

† Units in nanoseconds (ns).

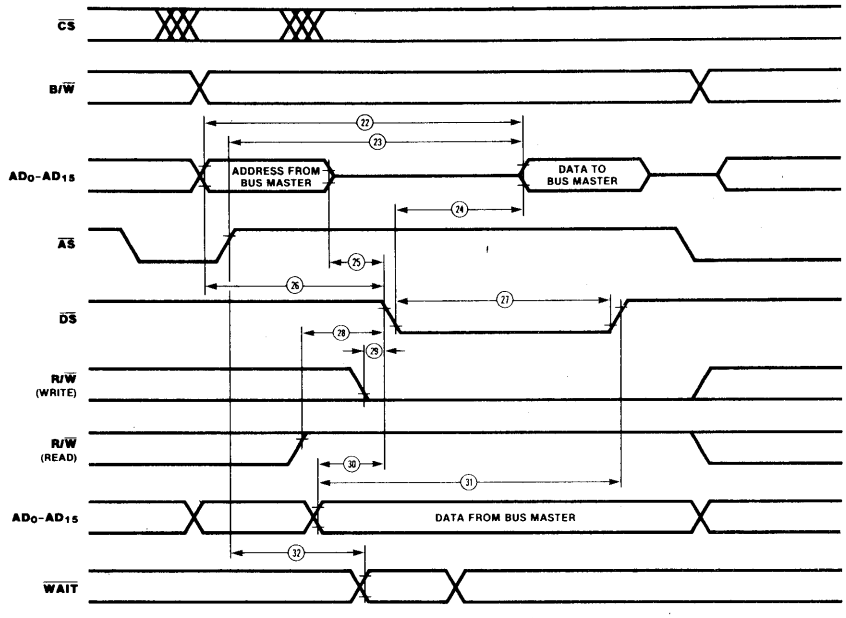
Except where otherwise stated, maximum rise and fall times for inputs are 200 ns.

Memory and Peripheral Timing

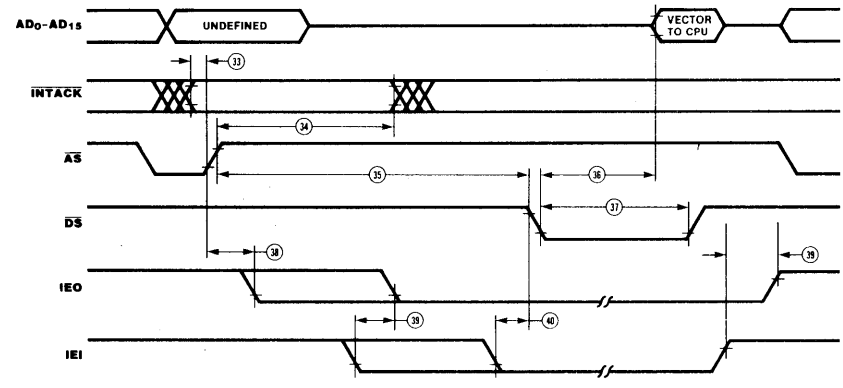


Parameters 1-12 are common to all transactions.

I/O Transaction Timing



Interrupt Acknowledge Timing



No.	Symbol	Parameter	4 MHz Min	4 MHz Max	6 MHz Min	6 MHz Max	Notes*†
		All Transactions					
1	TsCS(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Setup Time	0		0		1
2	ThCS(AS)	$\overline{CS}$ to $\overline{AS}$ ↑ Hold Time	60		40		1
3	TsS(AS)	Status to $\overline{AS}$ ↑ Setup Time	20		0		2
4	ThS(DS)	Status to $\overline{DS}$ ↑ Hold Time	55		40		
5	TsA(AS)	Address to $\overline{AS}$ ↑ Setup Time	30		10		1
6	ThA(AS)	Address to $\overline{AS}$ ↑ Hold Time	50		30		1
7	TwAS	$\overline{AS}$ Low Width	70		50	0	
8	TdDS(DR)	$\overline{DS}$ ↑ to Read Data Not Valid Delay	0				
9	TdDS(DRz)	$\overline{DS}$ ↑ to Read Data Float Delay		70	45		
10	TdAS(DS)	$\overline{AS}$ ↑ to $\overline{DS}$ ↓ Delay	60	2095	40		5
11	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	50		25		
12	ThDW(DS)	Write Data to $\overline{DS}$ ↑ Hold Time	30		20		1
		Memory Transactions					
13	TdA(DR)	Address Required Valid to Read Data Valid Delay		320		255	
14	TdAS(DR)	$\overline{AS}$ ↑ to Read Valid Delay		270		170	
15	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		0		
16	TdDSR(DR)	$\overline{DS}$ (Read) ↓ to Read Data Valid Delay		110		80	
17	TwDSR	$\overline{DS}$ (Read) Low Width	240		180		
18	TdA(DS)	Address to $\overline{DS}$ ↓ Setup	160		100		
19	TwDSW	$\overline{DS}$ (Write) Low Width	150		105		
20	TsDW(DSWf)	Write Data to $\overline{DS}$ (Write) ↓ Setup Time	30		20		
21	TsDW(DSWr)	Write Data to $\overline{DS}$ (Write) ↑ Setup Time	210		180		
		I/O Transactions					
22	TdA(DR)	Address Required Valid to Read Data Valid Delay		570		420	
23	TdAS(DR)	$\overline{AS}$ ↑ to Read Data Valid Delay		520		335	
24	TdDSI(DR)	$\overline{DS}$ (I/O) ↓ to Read Data Valid Delay		250		180	
25	TdAz(DSI)	Address Float to $\overline{DS}$ (I/O) ↓ Delay	0		0		
26	TdA(DSI)	Address to $\overline{DS}$ (I/O) ↓ Setup	160		100		
27	TwDSI	$\overline{DS}$ (I/O) Low Width	390		250		
28	TsRWR(DSI)	R/$\overline{W}$ (Read) to $\overline{DS}$ (I/O) ↓ Setup Time	100		100		
29	TsRWW(DSI)	R/$\overline{W}$ (Write) to $\overline{DS}$ (I/O) ↓ Setup Time	0		0		
30	TsDW(DSIf)	Write Data to $\overline{DS}$ (I/O) ↓ Setup Time	30		20		
31	TsDW(DSIr)	Write Data to $\overline{DS}$ (I/O) ↑ Setup Time	460		305		
32	TdAS(W)	$\overline{AS}$ ↑ to $\overline{WAIT}$ Valid Delay	195		160		
		Interrupt-Acknowledge Transactions					
33	TsIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Setup Time	0		0		
34	ThIA(AS)	$\overline{INTACK}$ to $\overline{AS}$ ↑ Hold Time	250		250		
35	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	940		675		
36	TdDSA(DR)	$\overline{DS}$ (Acknowledge) ↓ to Read Delay Valid Delay	365		245		
37	TwDSA	$\overline{DS}$ (Acknowledge) Low Width	475		310		
38	TdAS(IEO)	$\overline{AS}$ ↓ to IEO ↓ Delay					3,4
39	TdIEIf(IEO)	IEI to IEO Delay					4
40	TsIEI(DSA)	IEI to $\overline{DS}$ (Acknowledge) ↓ Setup Time					4

NOTES:
1. Parameter does not apply to Interrupt Acknowledge Transactions.
2. Does not cover R/$\overline{W}$ for I/O Transactions.
3. Applies only to a peripheral which is pulling $\overline{INT}$ Low at the beginning of the Interrupt Acknowledge Transaction.
4. These parameters are device dependent. The parameters for the devices in any particular daisy chain must meet the following constraint: for any two peripherals in the daisy chain, TdAS(DSA) must be greater than the sum of TdAS(IEO) for the higher priority peripheral, and TdIEIf(IEO) for each peripheral separating them in the daisy chain.
5. The maximum value for TdAS(DS) does not apply to Interrupt Acknowledge Transactions.

* Timings are preliminary and subject to change.
† Units in nanoseconds (ns).
Except where otherwise stated, maximum rise and fall times for inputs are 200 ns.

ZBI
Z-BUS Backplane Interconnect System

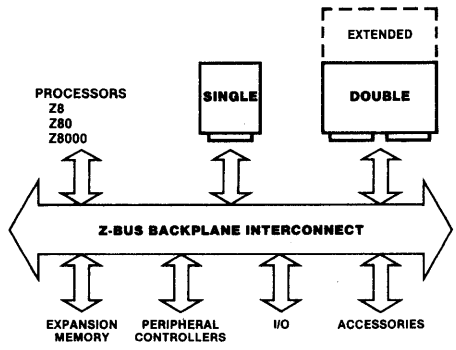
Zilog

Product Description

September 1983

Features

- The bus structure for the 80's
- Compatible with the Z-BUS Component Interconnect system
- Designed for the powerful Zilog Family of Microprocessors
  - Z8 CPU
  - Z80 CPU
  - Z8000 CPU
  - Future microprocessors
- Flexibility in application
  - 8-, 16- or 32-bit operations
  - Unsegmented, segmented, or memory mapped systems
- Future growth
  - Allows 32-bit operations
  - 5-bit status field
- Designed-in reliability
  - Byte-oriented parity and parity error line
  - High reliability pin and socket connectors
  - Distributed ground lines
  - High-current power distribution
  - Terminated bus lines



Description

The Z-BUS Backplane Interconnect (ZBI) system is a high performance, application-oriented system bus designed to utilize the full capabilities of all Zilog microprocessors—the Z8, Z80 and Z8000.

Thirty-two address/data lines coupled with twenty-eight control lines provide the resources needed for growth paths to future, more complex 32-bit microprocessors.

A member of the Z-BUS family of microcomputer bus structures, the ZBI bus is compatible with the Z-BUS Component Interconnect (ZCI) system used for communications at the chip level between Zilog processors and their peripheral support modules.

Reliability has been designed into the ZBI structure: parity lines have been included; ground lines are distributed between signals to reduce noise; and all bus lines are terminated.

Functional Description

Mechanical Configuration. The ZBI bus is defined for three sizes of modular boards. The single size modules measure 6.3" × 3.9" (160 mm × 100 mm). The double size boards are 6.3" × 9.2" (160 mm × 233.4 mm), and the double extended size measure 11.0" × 9.2" (280 mm × 233.4 mm). All ZBI boards are consistent with the standard European form factor.

The single size boards have a single bus connector while the double boards have two connectors.

The connector has a matrix of 96 pins on .100" (2.54 mm) centers which are aligned in 3 rows of 32 pins each. A molded plastic housing surrounds the pin array, providing mechanical rigidity and protecting the pins from mechanical damage.

The backplanes use a similar style of mating connector. These connectors are highly reliable because connection surfaces are completely enclosed and shielded from dirt and dust when the connectors are mated. Another advantage of this connector is its high density which permits the design of compact boards. In addition, the connectors are self-aligning

Functional Description (Continued)

and keyed to prevent improper insertion.

All of the signals on the double boards are assigned to one connector so that single boards can be used in the same backplane with double boards. The second connector on the double boards is unspecified and available for use by the designer.

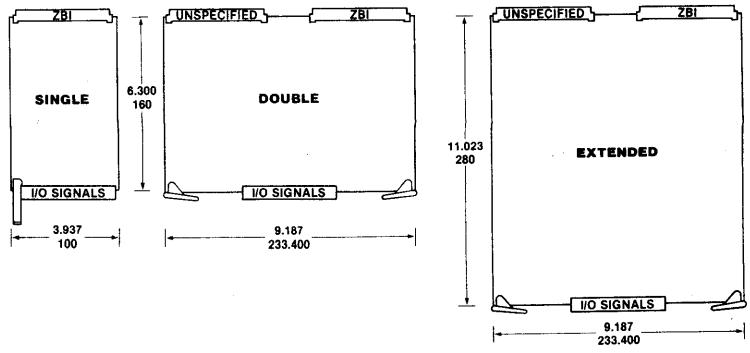


Figure 2. The ZBI backplane accepts two sizes of boards; both are compatible with European standards. The dimensions are in inches (upper) and millimeters (lower).

Signal Description

The ZBI consists of 96 lines: 32 bidirectional address/data lines with four parity lines, nine interrupt lines, 28 control lines, 21 power-supply lines for ±12 V, ±5 V and ground and two reserve lines. The pin layout was defined to provide the most convenient connection from the board and the backplane, with signals collected into logical groups for placement on the connector.

Address and Data. The address/data group is laid out to enable the lines to enter the board in order on both two-layer and four-layer boards. Low-order lines are placed next to the power pins for easier routing through buffers. The high-order address/data pins are positioned near the control pins to facilitate decoding of the state of the bus. Address and data information is transmitted over 32 bidirectional lines with separate address and data strobe lines arbitrating the information flow. The use of shared address/data lines enables a compact connection while still allowing 32-bit word sizes.

Word size is controlled by two lines that indicate the data width of the current operation on the bus, making possible 8-bit, 16-bit and 32-bit word transfers in the same system. Data is aligned in the lower byte (AD_0-AD_7) of the data field for 8-bit transfers, and the lower word (AD_0-AD_{15}) for 16-bit transfers.

The ZBI includes four parity lines and an error-indication line to detect errors in memory devices and transmissions on the bus. One parity bit is provided for each byte of the 32-bit address/data field, enabling parity-checking at both the byte and word levels.

Control Signals. The ZBI bus has 28 lines that are used for bus control and status, grouped into the following categories:

- *Clocking.* Two lines provide a master clock and a bus clock. The master clock supplies a constant frequency and is used as a master timing reference; the bus clock is derived from the master.
- *Extended processor architectures.* Two lines enable the CPU to interact with an Extended Processor Unit.
- *Resource sharing.* Three lines enable processors to lock other processors off the bus. This is a software implementation, and all processors must be aware of these signals for the lockout to be effective.
- *Direct memory access.* Three lines provide the control signals required for data to be transmitted in burst mode across the bus. When a DMA device wants to transmit information, it issues a request that causes the processor to get off the bus. Once off the bus, the processor issues an acknowledge signal indicating that the bus is free. The DMA device then begins transferring data to the specified address. When the transfer is complete, the processor regains use of the bus.
- *Muiltiprocessor.* Four lines enable multiple processors to share a common bus. (Arbitration logic to prevent contention errors must be included on each module.)
- *Data/Address Strobe.* Two lines indicate whether address or data information is on the address/data lines.
- *Status.* Five lines designate the kind of transaction occurring on the bus.
- *Word-size select.* Two lines determine the word size of the transaction on the bus.

Interrupts. The ZBI bus has three independent interrupt groups. Each group has an interrupt request line and an interrupt enable input and output daisy chain. A different priority level is assigned to each of the three interrupt groups and position-dependent priority is assigned to each device within the groups.

The treatment of the interrupt signal is processor-dependent and can be maskable, non-maskable, vectored, or non-vectored depending upon the configuration of the system CPU.

Bus Conditioning. All bus lines are terminated in resistor pairs to provide the highest integrity and best noise immunity for the system. This forces all undriven lines to approximately +3 V.

Signal Definition

Table 1 defines the signals necessary to the ZBI structure.

All signals, with the exception of the data and address lines, are negative true signals, where logical 1 = < 0.5 V and logical 0 = > +2.4 V. Any exception to this standard is noted in the table. Naming conventions are as follows:

NAME: a single line, negative-true logic level

NAME: a single line, positive-true logic level

NAME<0.3>: 4 lines, positive-true logic level

NAME1/NAME2: a doubly named line, High/Low logic levels

The abbreviations used to describe signal types are:

BD: Bidirectional data lines

TS: 3-state, undirectional lines

OC: Open collector

HC: High-current driver line, not 3-state

DC: Daisy-chained signal—OUT on one board connects to IN of the next board

Signal Name	Number of Lines	Signal Type	Function
Address and Data Group			
AD<0:31>	32	BD	**Address and Data Lines.** Address and data information is time-multiplexed onto these lines. The times they are valid are defined by address strobe ($\overline{AS}$) and data strobe ($\overline{DS}$). Additional information can be derived from the BCLK signal for synchronous operation.
Parity Group			
P<0:3>	4	BD	**Parity-Check Bits.** For bus transfer integrity, one parity bit is provided for each byte of the 32-bit address/data bus. Even parity ensures that a read from a non-existent resource will generate a parity fault.
$\overline{PE}$	1	OC	**Parity Error.** Indicates to the Bus Master that a parity error in a data transfer on the bus has been caught by the parity check logic.
Interrupt Group			
$\overline{INT1}$	1	OC	**Level 1 Interrupts.** Highest priority interrupt in the system. If a non-maskable interrupt is present, it must be here.
$\overline{INT2}$	1	OC	**Level 2 Interrupt.** Second highest priority interrupt in the system. If a vectored interrupt is present, it must be here.
$\overline{INT3}$	1	OC	**Level 3 Interrupt.** Lowest priority interrupt in the system. If a non-vectored interrupt is present, it must be here.
IEI1	1	DC	Level 1 Interrupt Enable In
IEO1	1	DC	Level 1 Interrupt Enable Out
IEI2	1	DC	Level 2 Interrupt Enable In
IEO2	1	DC	Level 2 Interrupt Enable Out
IEI3	1	DC	Level 3 Interrupt Enable In
IEO3	1	DC	Level 3 Interrupt Enable Out
Control Group			
$\overline{PWRBAD}$	1	OC	**Power Bad.** An early warning signal that the dc power for the system will soon disappear. This signal is generated by the power supply to give the processor enough time to store the machine state (if appropriate storage is available) before power drops below critical levels.
Clocking			
MCLK	1	HC	**Master Clock.** System master clock—16 to 32 MHz. Frequency is a 4× multiple of the desired bus clock frequency.
BCLK	1	HC	**Bus Clock.** Bus transaction clock, derived from Master Clock and used by all synchronous elements in the system.
Extended Processing Architecture			
N/$\overline{S}$	1	TS	**Normal/System.** Indicates the mode of the CPU controlling the bus—Normal user mode or System mode (able to execute privileged instructions).
$\overline{STOP}$	1	OC	**Stop Line.** Stop the processor in control of the bus for synchronization of activities with the CPU.
Address/Data Strobes			
$\overline{AS}$	1	TS	**Address Strobe.** Indicates that the AD lines contain a valid address. The $\overline{AS}$ line is pulsed low by a board controlling the transaction for program or data memory access. Addresses are valid at the trailing (rising) edge of $\overline{AS}$.
$\overline{DS}$	1	TS	**Data Strobe.** Data is placed on or accepted from the AD bus lines when $\overline{DS}$ is low.

Table 1. Signal Definitions

Signal Name	Number of Lines	Signal Type	Function
Status			
ST<0:4>	5	TS	**Status Lines.** These lines designate the type of transaction occurring on the bus.

S_4	S_3	S_2	S_1	S_0	Transaction
0	0	0	0	0	Internal Operation
0	0	0	0	1	Memory Refresh
0	0	0	1	0	I/O Reference
0	0	0	1	1	Special I/O Reference
0	0	1	0	0	Segment Trap Ack
0	0	1	0	1	Int1 Interrupt Ack
0	0	1	1	0	Int2 Interrupt Ack
0	0	1	1	1	Int3 Interrupt Ack
0	1	0	0	0	Data Memory Request
0	1	0	0	1	Stack Memory Request
0	1	0	1	0	Data Mem <> EPU transfer
0	1	0	1	1	Stack Mem <> EPU transfer
0	1	1	0	0	Prog Ref - nth cycle
0	1	1	0	1	Prog Ref - 1st cycle
0	1	1	1	0	EPU <> CPU transfer
0	1	1	1	1	Reserved
1	X	X	X	X	Reserved

Signal Name	Number of Lines	Signal Type	Function
Word Size Select			
B/$\overline{W}$	1	TS	**Byte/Word Select.** Used in conjunction with W/$\overline{LW}$ to define data access width.
W/$\overline{LW}$	1	TS	**Word/Long Word Select.** Used in conjunction with B/$\overline{W}$ to define the data access width. (A logical 1 is a high voltage level.)

B/$\overline{W}$	W/$\overline{LW}$	Access Width
1	1	Byte (8-bit)—Data on AD <0:7>
0	1	Word (16-bit)—Data on AD <0:15>
1	0	Double Word (32-bit)—Data on AD <0:31>
0	0	Reserved

Resource Sharing			
$\overline{MMREQ}$	1	OC	**Multimicro Request.** This is a software request to another processor for software synchronization.
$\overline{MMAI}$	1	DC	**Multimicro Acknowledge In.** Forms the logical chain among processors to perform software arbitration, in conjunction with the $\overline{MMAO}$ signal. The effect of this line is dependent on the software present on the processor board.
$\overline{MMAO}$	1	DC	**Multimicro Acknowledge Out.** Completes the logical chain to the next processor's $\overline{MMAI}$ pin.
Direct Memory Access			
$\overline{BAI}$	1	DC	**Bus Acknowledge In From Priority Chain.** This signal and $\overline{BAO}$ form the bus priority chain.
$\overline{BAO}$	1	DC	**Bus Acknowledge Out to Priority Chain.** Completes the circuit to the next device in the bus priority chain.
$\overline{BUSREQ}$	1	OC	**Bus Request.** Used to request access to the bus. A request to a processor to relinquish the bus at the end of the current instruction cycle. This signal is used with the $\overline{BAI}$ and $\overline{BAO}$ signals to control bus sharing by DMA devices not able to become bus masters.
Multiprocessor Control			
$\overline{CAI}$	1	DC	**CPU Acknowledge In.**
$\overline{CAO}$	1	DC	**CPU Acknowledge Out.**
$\overline{CPUREQ}$	1	DC	**CPU Request.** A request to the processor currently in control of the bus to relinquish control at the end of the current instruction cycle. This signal is used with $\overline{CAI}$, $\overline{CAO}$, and CAVAIL to control sharing of the bus by devices able to become bus masters.
CAVAIL	1	TS	**CPU Available.** Used in conjunction with $\overline{CAI}$ and $\overline{CAO}$ to transfer bus control from one bus master to another.
Miscellaneous Control Lines			
$\overline{RESET}$	1	OC	**Reset.** Connected to the master reset switch and power-up reset circuit.
$\overline{WAIT}$	1	OC	**Wait.** Causes a processor or peripheral to wait for the response to a request for data. Such a wait could be caused by slow memory or by refresh contention problems.
R/$\overline{W}$	1	TS	**Read/Write.** If this line is high, the current operation is a read; if low, a write.

Table 1. Signal Definitions (continued)

Zilog

Military Specification Standards

June 1982

General Description

Zilog offers high-reliability versions of the entire family of Z8, Z80, and Z8000 logic circuits, processed in accordance with the requirements of MIL-STD-833 (Test Methods and Procedures for Microelectronics). The Z80 and Z8000 CPUs are currently Qualified Products, Level II MIL-M-38510.

General Considerations. Zilog high-reliability microcircuits are designed to meet the full military temperature range of -55°C to +125°C and are packaged in hermetic dual-in-line packages. These packages can reliably withstand the thermal shock requirements of MIL-STD-833, method 1011, Condition C (-65°C to +150°C). For industrial users, Zilog offers an extended operating temperature range of -40°C to +85°C. All of Zilog's high-reliability microcircuits receive 5004 processing in accordance with the requirements of MIL-STD-833. Table 1 lists the screening tests performed on the two levels. An X indicates that the test is performed 100% of the time. Additional screening options are available upon request. Table 2 lists the Zilog products available with the 100% testing process shown with X's in Table 1.

Test	Condition	MIL-STD-883 Method	MIL-STD-883 Condition	Class B	Class C
Precap Visual	—	2010	B	X	X
Seal and Lot I.D.	—	—	—	X	X
Stabilization Bake	48 hrs. @ 150°C	1008	C	X	X
Temperature Cycling	10 cycles	1010	C	X	X
Centrifuge	Y_1 Plane	2001	E	X	X
Fine Leak	—	1014	A	X	X
Gross Leak	—	1014	C	X	X
Electrical Test	Per Zilog Data Sheets	—	—	X	X
Burn-In	160 hr. + 8 - 0	1015	160 hrs.	X	—
Final Electrical	-55°C,	—	—	X	X
	and +125°C	—	—	X	X
External Visual	—	2009	—	X	X

NOTES: S = Sample testing only, X = 100% testing.

Table 1. Total Lot Screening

General Description (Continued)

Product*	Speed	883 Temp Range	Extended Temp Range
Z8611	8.0 MHz	Yes	Yes
Z8681	8.0 MHz	Yes	Yes
Z80 CPU	2.5 MHz	Yes	Yes
Z80A CPU	4.0 MHz	Yes	Yes
Z80 PIO	2.5 MHz	Yes	Yes
Z80A PIO	4.0 MHz	Yes	Yes
Z80 SIO	2.5 MHz	Yes	Yes
Z80A SIO	4.0 MHz	Yes	Yes
Z80 CTC	2.5 MHz	Yes	Yes
Z80A CTC	4.0 MHz	Yes	Yes
Z8001 CPU	4.0 MHz	Yes	Yes
Z8001A CPU	6.0 MHz	Yes	Yes
Z8002 CPU	4.0 MHz	Yes	Yes
Z8002A CPU	6.0 MHz	Yes	Yes
Z8030 Z-SCC	4.0 MHz	Yes	Yes
Z8030A Z-SCC	6.0 MHz	Yes	Yes
Z8036 CIO	4.0 MHz	Yes	Yes
Z8036A CIO	6.0 MHz	Yes	Yes
Z8038 Z-FIO	4.0 MHz	Yes	Yes
Z8038A Z-FIO	6.0 MHz	Yes	Yes
Z8530 SCC	4.0 MHz	Yes	Yes
Z8530A SCC	6.0 MHz	Yes	Yes
Z8536 CIO	4.0 MHz	Yes	Yes
Z8536A CIO	6.0 MHz	Yes	Yes

*NOTE: See Ordering Information for package and temperature designators.

Table 2. High-Reliability Products Available

Manufacturing and Process Controls

Zilog high-reliability microcircuits will be processed and assembled in accordance with the requirements of Appendix A of MIL-M-38510 or MIL STD 883, as specified by customer purchase order. The following are some of the items contained in the Zilog Product Assurance Program Plan:

- A clear, concise procedure for converting a customer specification to a Zilog internal specification, assuring the customer that parts received meet or exceed specified requirements.
- A formalized training and testing program for operators and inspection personnel to ensure that each operation is performed correctly.
- An inspection system that includes a complete Incoming Inspection Laboratory and a Chemical Analysis Laboratory ensure that all materials, utilities, and work-in-progress meet Zilog requirements and specifications.
- Rigid requirements for the cleanliness of work areas and the maintenance of a Class 100 environment at all stations where critical operations are performed.
- A document control system to control changes in design, materials, and processes.
- An instrument maintenance and calibration program complying to the requirements of MIL-STD-45662 (Calibration System Requirements).
- A quality audit system in accordance with MIL-I-45208 (Quality Program Requirements).

Zilog offers a number of standard flows, which include both military and commercial temperature ranges (see Table 3).

Environmental Flow	A	B	B1	G	H	K	Q
Temperature Range	S,E	M	M	S,E	S,E	S,E	M
Package	C,D,L	C,D,L	C,D,L	D,P	C,D,L,P	P	C,L
Pre-Cap Visual Method 2010, Condition B	X	X	X	X	X	X	X
Stabilization Bake Method 1008, Condition C	X	X	X	X	X	X	X
Temperature Cycle Method 1010, Condition C	X	X	X	X		X	X
Centrifuge Method 2001, Condition E (y, axis only)		X	X				X
Fine and Gross Leak Method 1014, Condition Fine A_2 Gross C	X	X	X				X
100% Electrical Test	X	X	X	X	X	X	X
Burn-In Method 1015, Condition D	X	X	X	X	X		X
100% Electrical Test	X	X	X	X	X	X	X
QA Test at Temperature	X	X	X	X	X	X	X
100% Electrical Test	X	X	X	X	X	X	X
Group B/C/D Generic Data		X					X

Note: All test methods are per MIL-STD-883.

Table 3. Standard Flows

Packaging Information

Zilog

Pioneering the Microworld

Zilog

September 1983

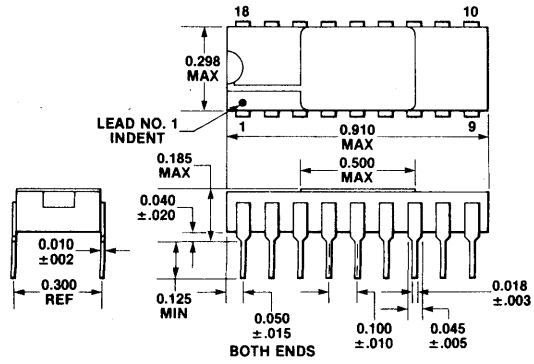
Package Information

This table summarizes the microprocessor components available from Zilog by number of pins and package type. Following the table are detailed drawings for each package type. For further information on specific components, see the Ordering Information section of each product specification.

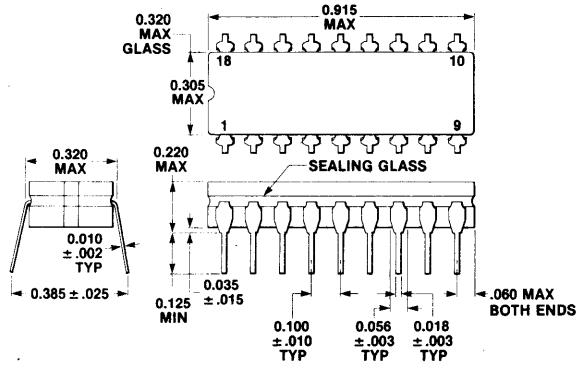
Pins	Package	Component
18	Ceramic, Cerdip, Plastic	Z8581 CGC
28	Ceramic, Cerdip, Plastic	Z8430 Z80 CTC
28	Leadless Carrier, Ceramic	Z8430 Z80 CTC
40	Ceramic, Cerdip, Plastic	Z8002 Z8000 CPU Z8030 Z8000 Z-SCC Z8036 Z8000 Z-CIO Z8038 Z8000 Z-FIO Z8090 Z8000 Z-UPC Z8400 Z80 CPU Z8410 Z80 DMA Z8420 Z80 PIO Z8440 Z80 SIO/0 Z8441 Z80 SIO/1 Z8442 Z80 SIO/2 Z8449 Z80 SIO/9 Z8470 Z80 DART Z8530 SCC Z8536 CIO Z8538 FIO Z8590 UPC Z8601 Z8 MCU Z8611 Z8 MCU Z8671 Z8 MCU Z8681 Z8 MCU
40	Protopack	Z8093 Z8000 Z-UPC Z8094 Z8000 Z-UPC Z8593 UPC Z8594 UPC Z8603 Z8 MCU Z8613 Z8 MCU
44	Leadless Carrier, Ceramic	Z8002 Z8000 CPU Z8030 Z8000 Z-SCC Z8036 Z8000 Z-CIO Z8038 Z8000 Z-FIO Z8400 Z80 CPU Z8410 Z80 DMA Z8420 Z80 PIO Z8444 Z80 SIO* Z8530 SCC Z8536 CIO
48	Ceramic, Plastic	Z8001 Z8000 CPU Z8010 Z8000 Z-MMU
52	Leadless Carrier, Ceramic	Z8010 Z8000 Z-MMU Z8001 Z8000 CPU
64	Ceramic, Plastic	Z8015 Z8000 Z-PMMU Z8116 Z800 MCU Z8216 Z800 MCU Z8612 Z8 DM
68	Leadless Carrier, Ceramic	Z8015 Z8000 Z-PMMU Z8070 APU Z8116 Z800 MCU Z8216 Z800 MCU Z8612 Z8 DM

*NOTE: As a result of size of package, all three SIO versions are included in one version, the Z8444.

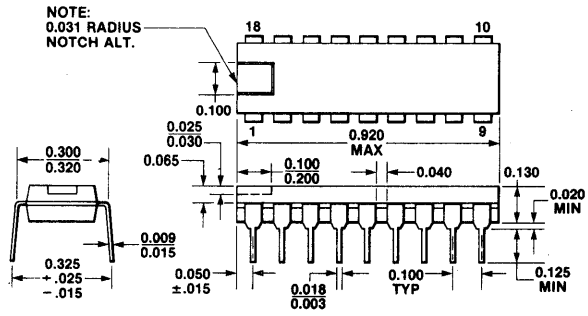
Package Information (Continued)



18-Pin Ceramic Package

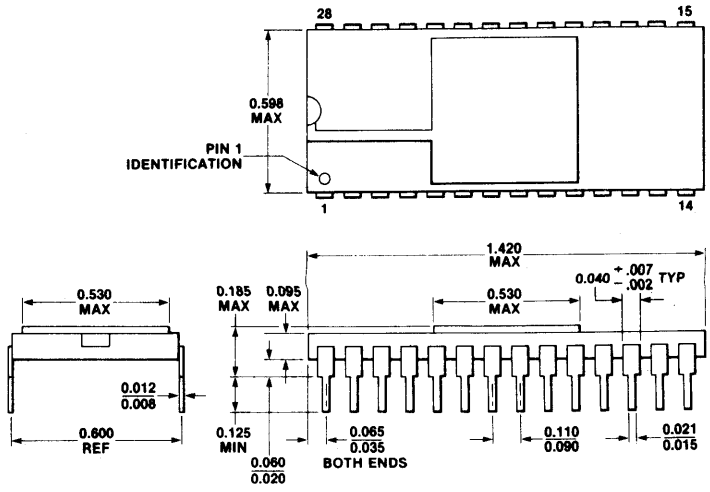


18-Pin Cerdip Package

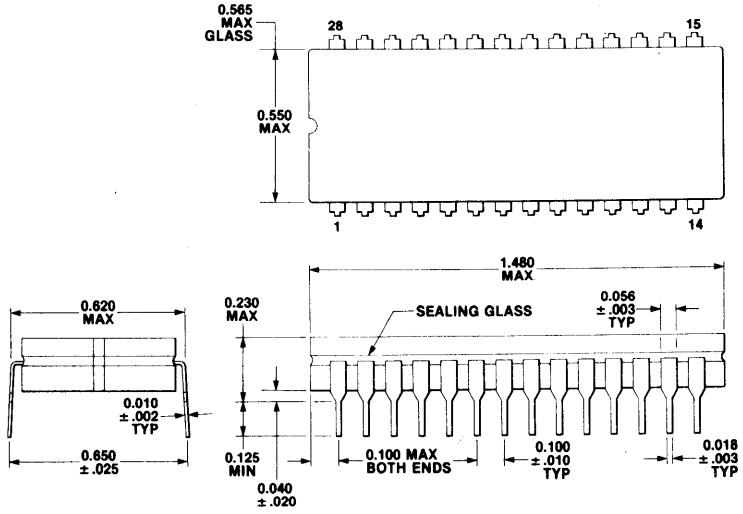


18-Pin Plastic Package

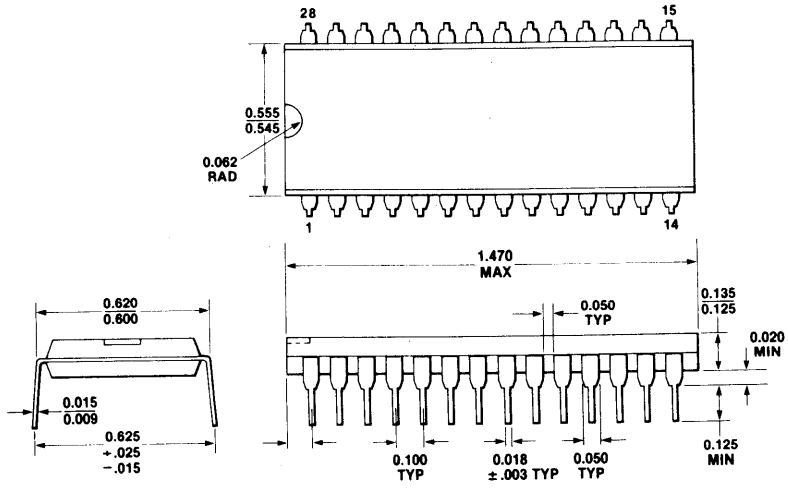
NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.



28-Pin Ceramic Package

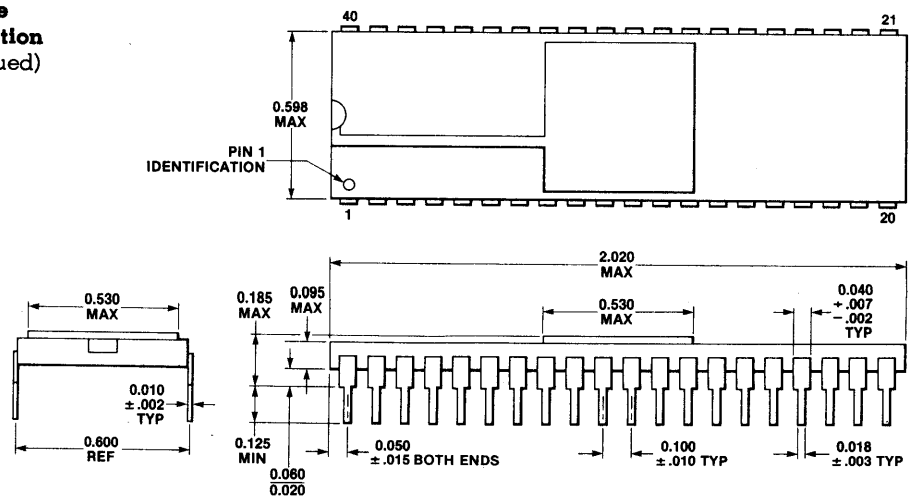


28-Pin Cerdip Package

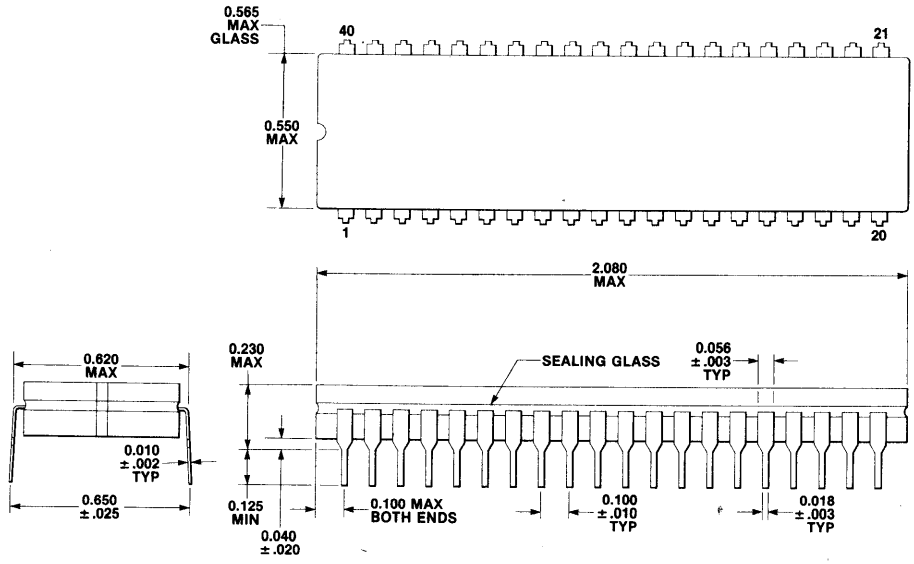


28-Pin Plastic Package

Package Information (Continued)



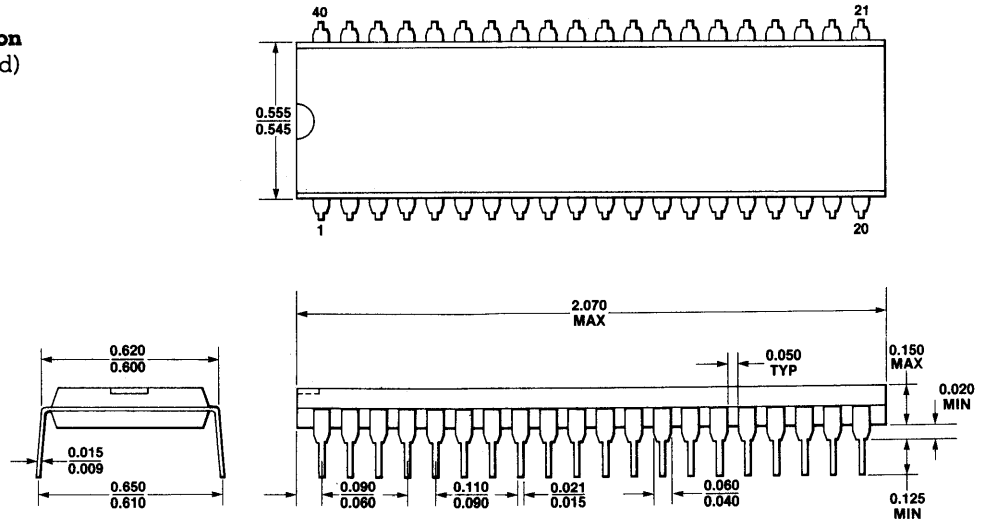
40-Pin Ceramic Package



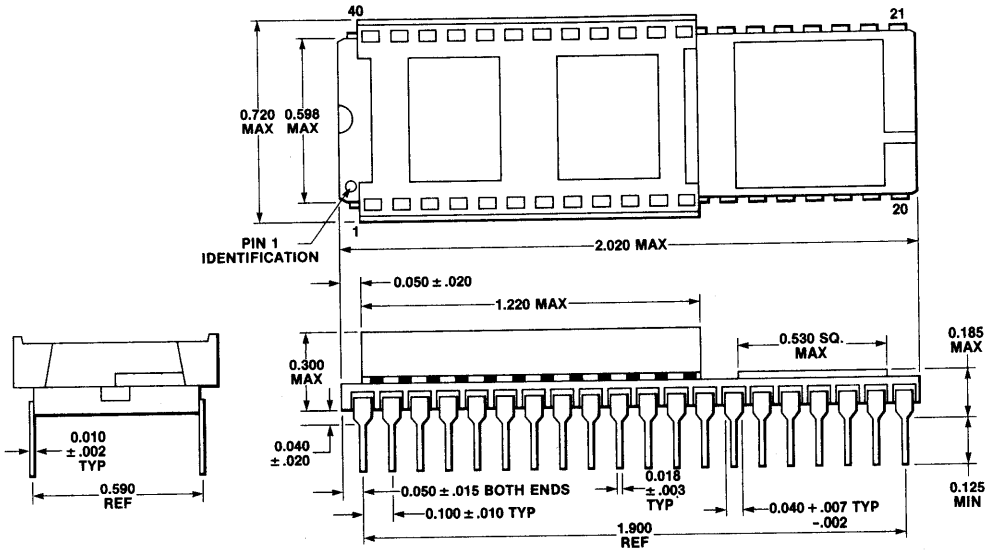
40-Pin Cerdip Package

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

Package Information (Continued)

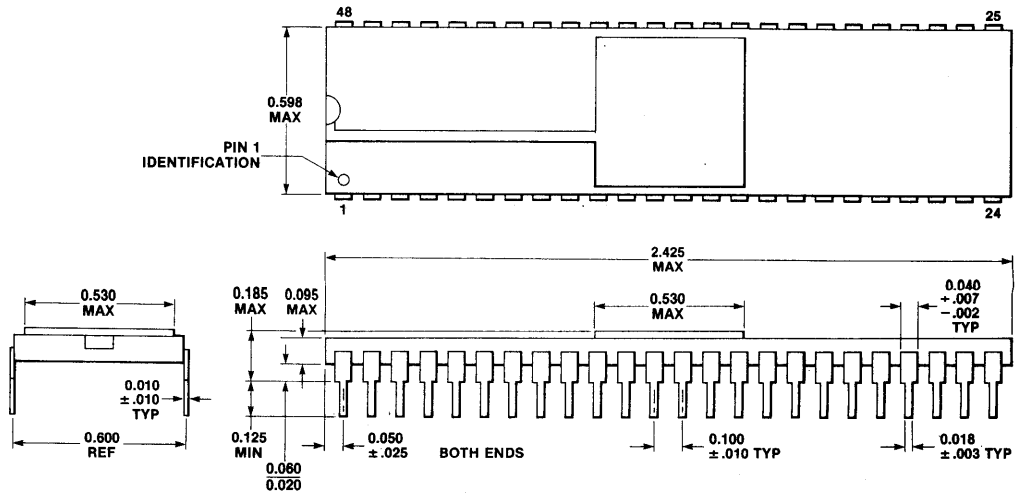


40-Pin Plastic Package

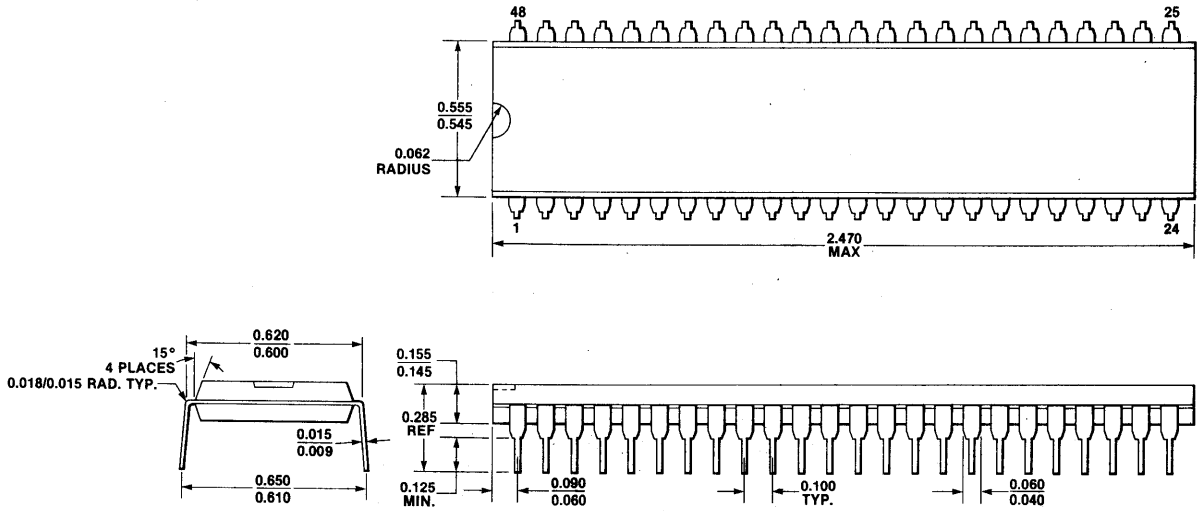


40-Pin Protopack Package

Package Information (Continued)



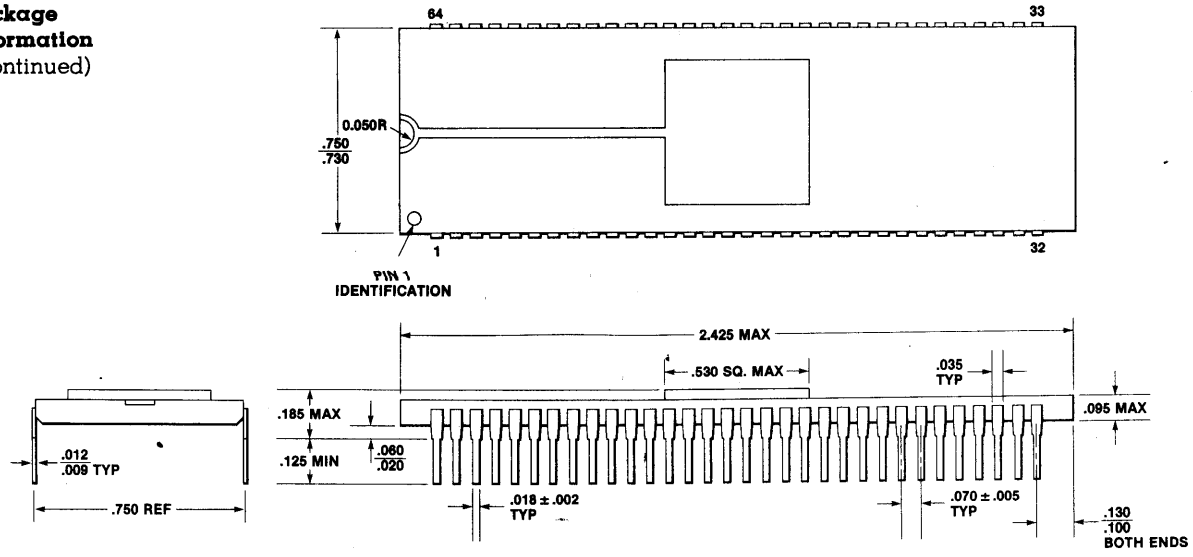
48-Pin Ceramic Package



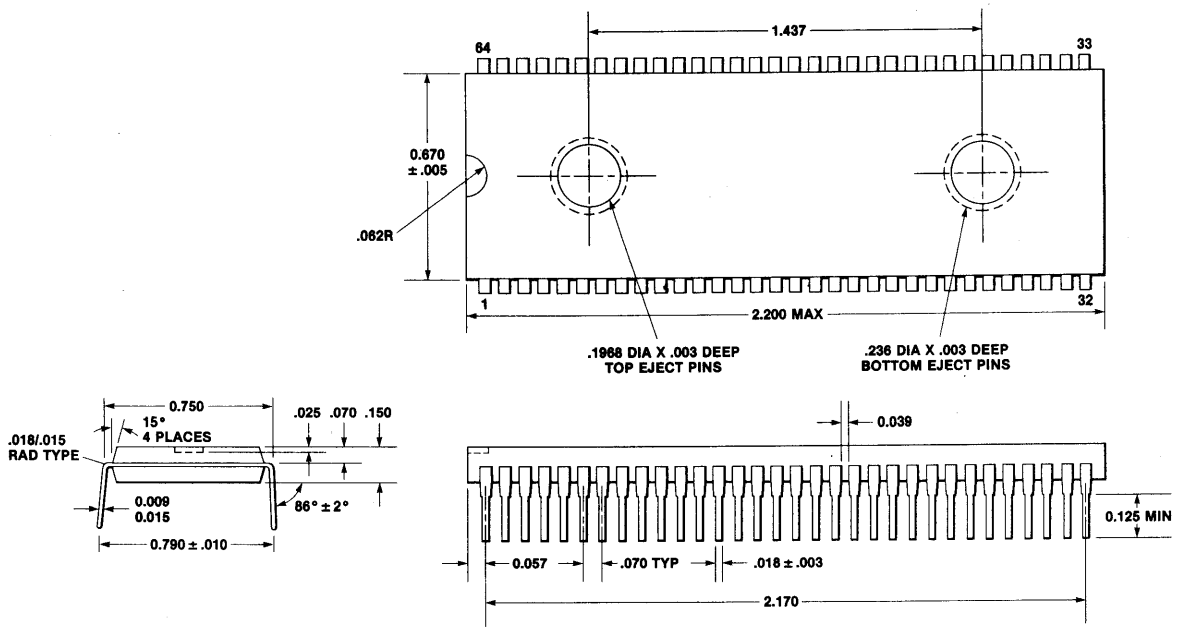
48-Pin Plastic Package

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

Package Information (Continued)

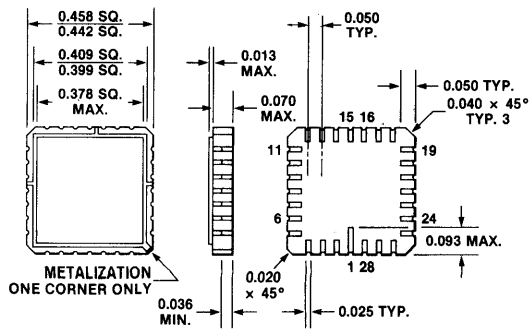


64-Pin Ceramic Package

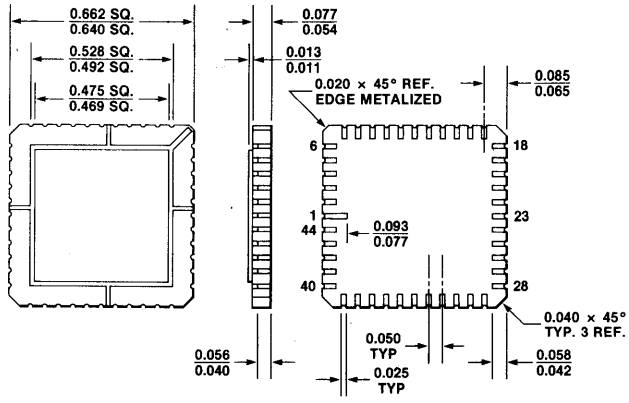


64-Pin Plastic Package

Package Information (Continued)

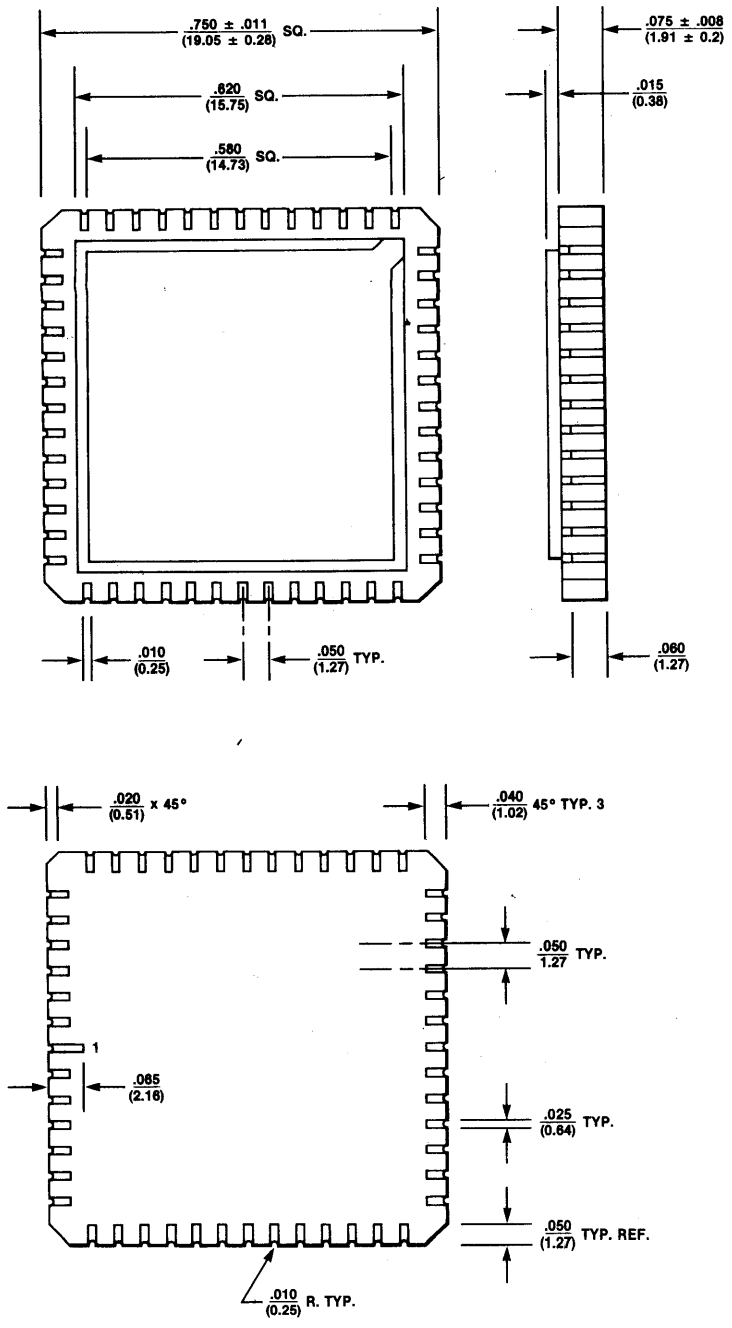


28-Pin Leadless Package



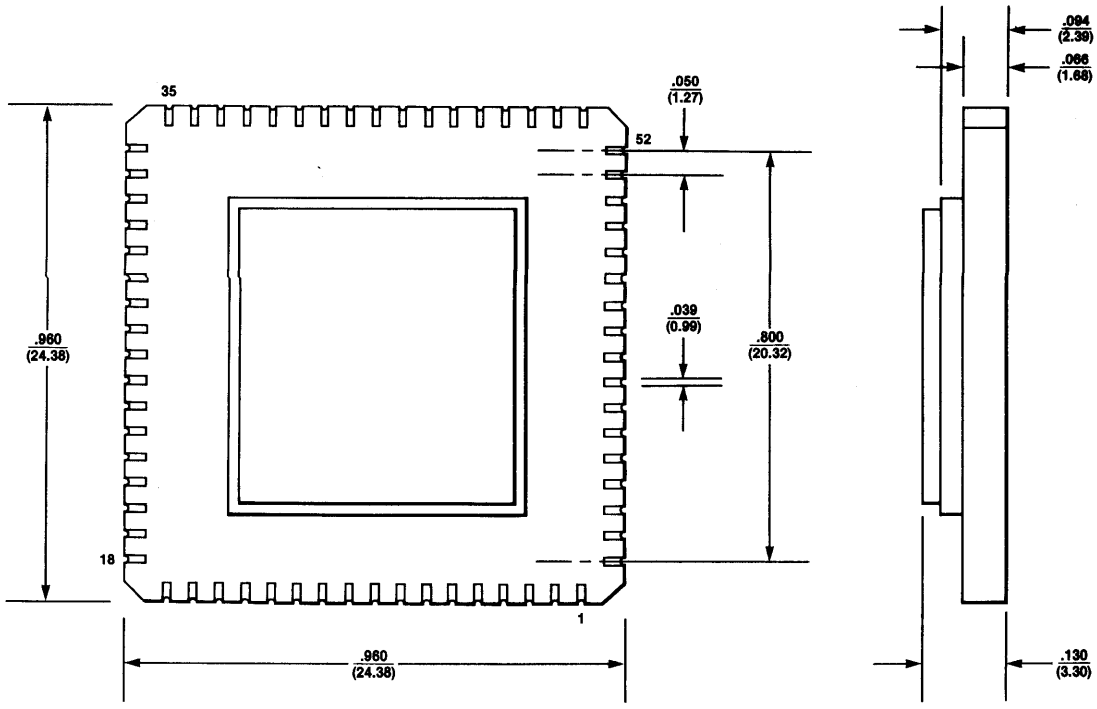
44-Pin Leadless Package

NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

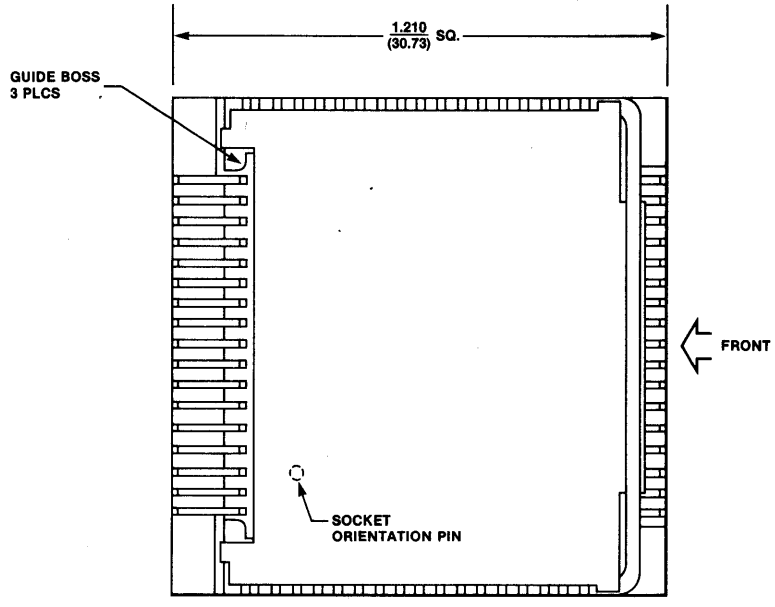


52-Pin Leadless Ceramic Package

Package Information (Continued)



JEDEC Type A Package

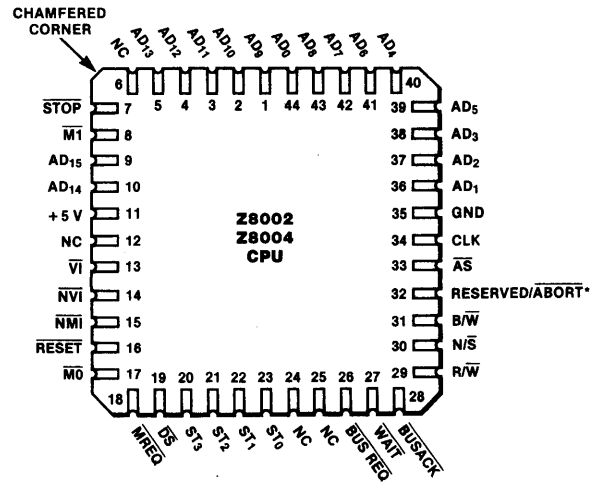


Textool Carrier Socket

68-Pin Leadless Ceramic Package

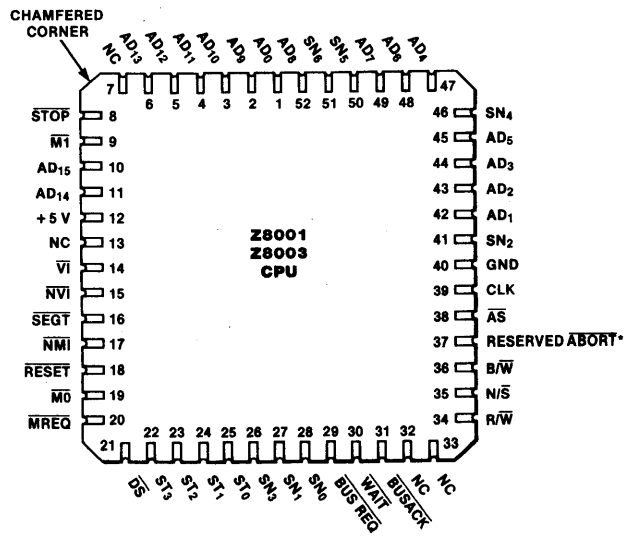
NOTE: Package dimensions are given in inches. To convert to millimeters, multiply by 25.4.

Package Information (Continued)



* = Z8004 Version
NC = No connection

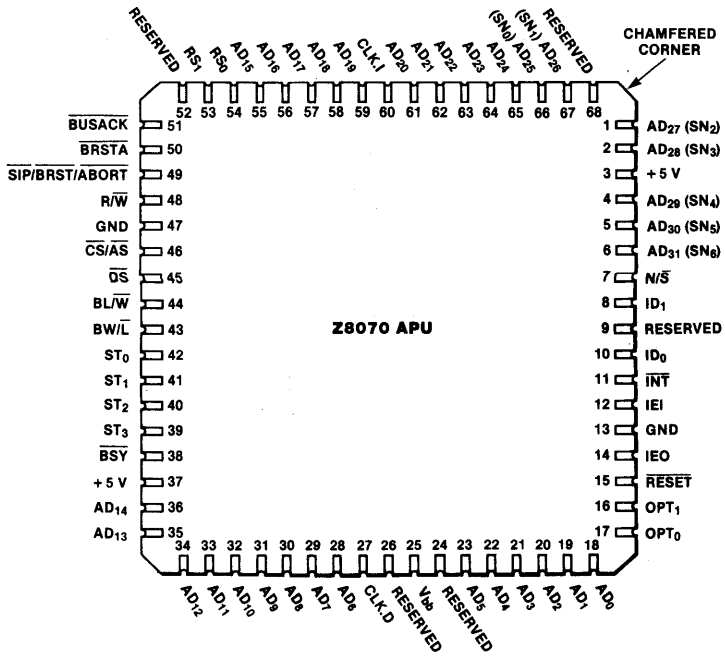
Z8002/4 Leadless Package



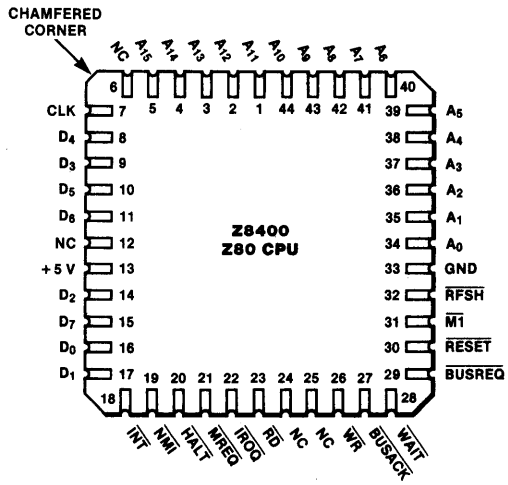
* = Z8003 only
NC = No connection

Z8001/3 Leadless Package

Package Information (Continued)



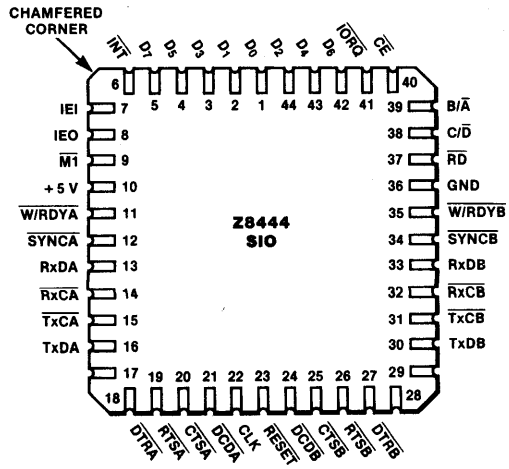
Z8070 APU Leadless Package



NC = No connection

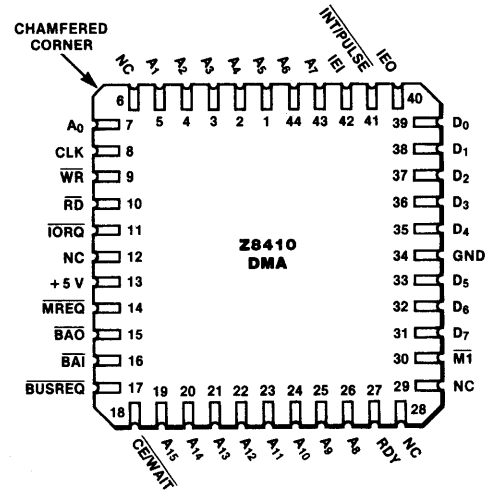
Z8400 Z80 CPU Leadless Package

Package Information (Continued)



NC = No connection

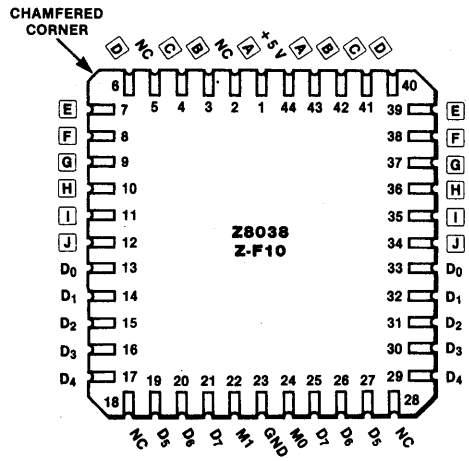
Z8444 SIO Leadless Package



NC = No connection

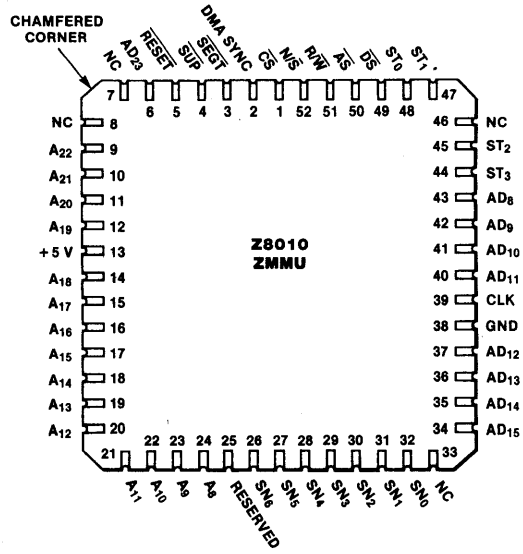
Z8410 DMA Leadless Package

Package Information
(Continued)



NC = No connection

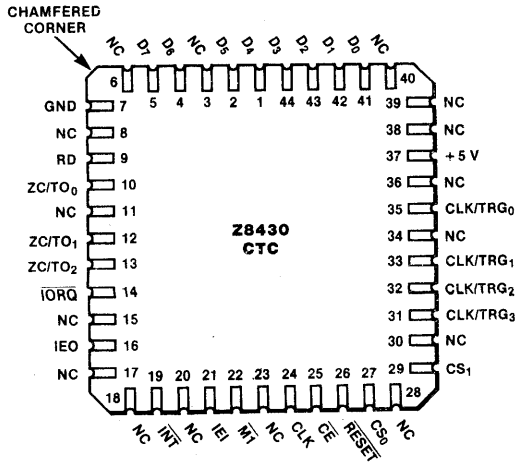
Z8038 Z-FIO Leadless Package



NC = No connection

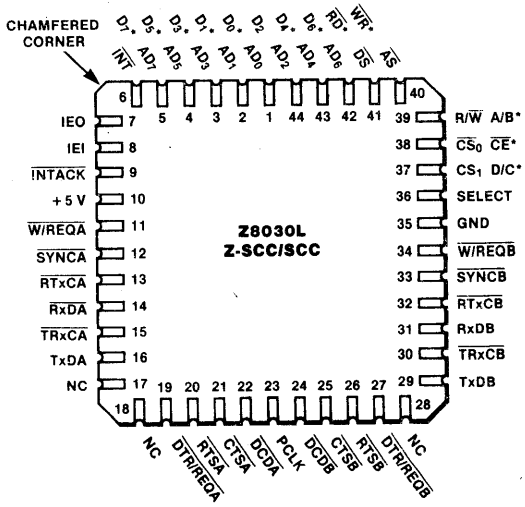
Z8010 Z-MMU Leadless Package

Package Information (Continued)



NC = No connection

Z8430 CTC Leadless Package

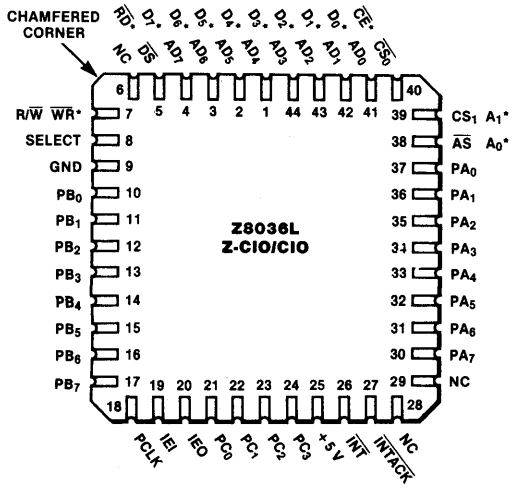


* = Z8530 Version
NC = No connection

Both Z8030 and Z8530 are implemented in a single leadless package and controlled by select pin 36 requiring: 5 volts for Z8030; GND for Z8530.

Z8030L Z-SCC/SCC Leadless Package

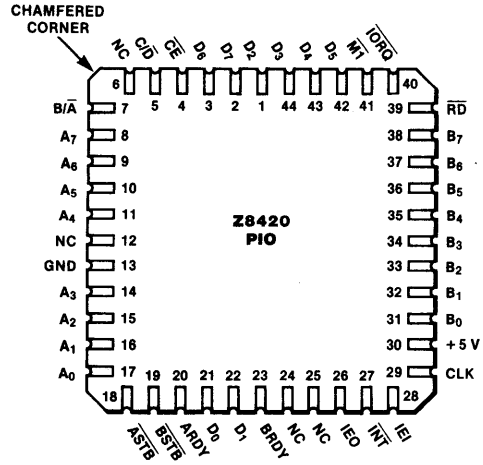
Package Information (Continued)



* = Z8536 Version
NC = No connection

Both Z8036 and Z8536 are implemented in a single leadless package and controlled by select pin 8 requiring: 5 volts for Z8036; GND for Z8536.

Z8036L Z-CIO/CIO Leadless Package



NC = No connection

Z8420 PIO Leadless Package

Development Products

Zilog

Pioneering the Microworld

Comprehensive Development Environments for All Zilog Microprocessors

Zilog's development system products feature ideal environments for software development for the Z8 and Z8000 microprocessors. The modularized design approach of the Zilog development systems allows the user a choice of hardware and software modules to meet current needs, while providing the necessary upgrade possibilities for future requirements.

The System 8000 concept partitions software and hardware development tools into specially tailored devices. Software and hardware checkout are handled by separate yet compatible products. Software can be developed on both Zilog and non-Zilog hosts using available compilers and cross-compilers. In either case, compatible hardware emulation systems are available at several levels of complexity. Standard RS-232 links provide for uploading and downloading of programs between hosts and emulators.

System 8000, a high-performance, multiuser, multitasking software development host combines the commercial system's function and the development system concept. The 6 MHz Z8000-based System 8000 hardware incorporates a high-performance Winchester disk, as well as intelligent disk and tape controllers, to further improve performance. ZEUS, the UNIX*-based operating system is specifically designed for software development and text processing. Numerous development tools are available, including the programming languages PLZ/SYS, C, FORTRAN 77, and Pascal; various libraries; and a symbolic debugger. Because ZEUS treats emulators as System 8000 peripherals, System 8000 can combine with EMS 8000, Z-SCAN 8000, Z-SCAN 8, or non-Zilog emulators to provide total product development support for multiple microprocessors.

Zilog has a stand-alone, low-level emulation device called Z-SCAN, and a high-level emulation device called EMS 8000. The basic idea behind Zilog development systems is to allow hardware and software to be developed simultaneously from the beginning of the project. Along with Zilog's System 8000 host (or other UNIX hosts, such as VAX or PDP 11), a multiuser development environment can be created in conjunction with the Z-SCAN family and the EMS 8000.

The Z-SCAN family includes Z-SCAN 8000 and Z-SCAN 8, with future plans for Z-SCAN UPC and Z-SCAN 800. Z-SCAN (Zilog Stand-Alone Circuit Analyser) is designed to be easy to master and so low-cost that every engineer can afford one.

The first in the family of the Z-SCANs is Z-SCAN 8000, which provides in-circuit emulation for both Z8001 and Z8002. It includes user-friendly, screen-oriented software and one complex breakpoint. Z-SCAN 8 has the same features as the Z-SCAN 8000; it also includes two hardware breakpoints and a real-time trace. Future Z-SCANs will provide additional features while still maintaining the Z-SCAN's user-friendly interface and low cost.

EMS 8000 is a sophisticated emulation management system that aids in the development of Z8000 implementations. By providing logic state analysis, high-speed emulations (up to 6 MHz), complex triggering, a large real-time trace buffer, and large mappable memory, EMS 8000 makes emulation and debugging both easier and faster. EMS 8000 also provides in-circuit emulation for Z8001, Z8002, and Z8003.

The Z8 and Z8000 Development Modules are single-board microcomputers that permit the development of code for the Z8, Z8001, or Z8002. They facilitate prototyping with large wire-wrap areas and are totally transparent to the CRTs and host CPU systems.

*UNIX is a trademark of Bell Laboratories.

Z8000™ Dual-Processor System Upgrade Package

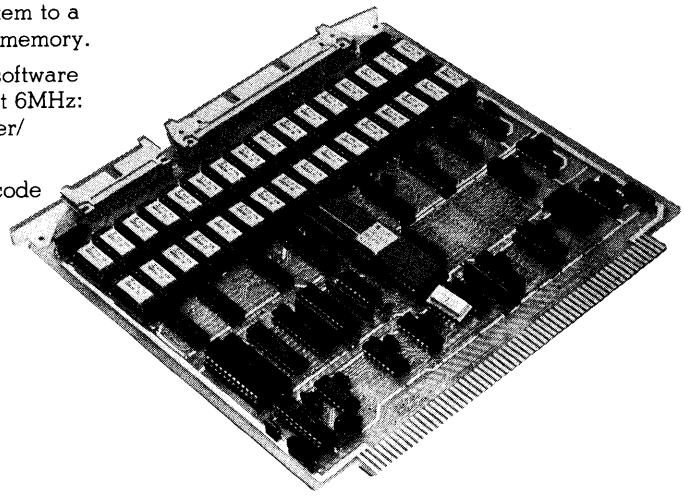
Zilog

Product Description

September 1983

Features

- Upgrades an MCZ™ or PDS system to a 16-bit system with 256K bytes of memory.
- Provides complete Z8000-based software development tools that execute at 6MHz: screen editor, translator, compiler/assembler, debugger.
- All software supplied in source code form to allow customization for applications.
- Existing Z80 programs continue to run.



Overview

The Z8000 Dual-Processor System Upgrade Package provides 16-bit processing power, 256K bytes of random access memory, and software development tools for Zilog's Z80-based MCZ and PDS systems. The package consists of a Z8000 Microprocessor Board (MPB/256), a screen editor, Z80 to Z8000 translator, and a Z8000 assembler and debugger.

The Z8000 MPB plugs directly into the backplane of the Z80 system and works with the Z80 and RIO. When the Z8000 is running, the Z80 acts as a peripheral processor that manages the resources of the host system. When the Z8000 is not activated, the system is controlled by the Z80 and there is no functional change due to the additional Z8000 MPB.

Functional Description

Hardware. The Z8000 MPB contains a 6MHz Z8001 CPU and 256K bytes of RAM. The board uses a FIFO for inter-CPU block transfers. The FIFO is implemented using control logic and a 1Kx8 static RAM chip and can be accessed sequentially by either the Z80 or the Z8000. Software resolves any contention for ownership.

The MPB can be plugged directly into the top slot of an MCZ-1/05 with no modification to the system. Or, it can be used in any vacant slot of an MCZ-1/20 or PDS 8000 system, and may require minor modification to the backplane.

Software. The software tools provided with the Z8000 Dual-Processor System Upgrade Package include a screen editor, Y (a multi-level language compiler), a symbolic debugger, a Z80 to Z8000 translator, and the interface software between the Z80 and Z8000.

The screen editor takes advantage of the 6MHz Z8000 and 256K RAM to provide an easy-to-use, efficient means of entering and modifying programs. The screen provides a "window" over the current copy of the file in memory. The cursor may be moved anywhere in the window to indicate the position where characters are to be added, deleted or

Functional Description (Continued)

replaced. In addition, commands are available to find and change strings of text and to delete, move or copy blocks of text. The screen editor is designed to be used with an Infoton 200 or a Visual 200 terminal. It can, however, be modified to work with almost any CRT.

The compiler, Y, is a multi-level language. It includes Z8000 assembly language with Zilog mnemonics, Pascal-like control structures, data types, arithmetic expressions with automatic or specified allocation of registers, procedure calls with parameter passing, and a descriptive compiler language. The different levels may, for the most part, be freely mixed. The Y compiler features direct, one-pass code generation into memory, immediate execution of statements, conditional compilation, user-defined language extensions and symbolic debugging.

The debugger can operate in two modes: Debug and Command. With the symbolic debugger in Debug mode, any instruction typed is executed immediately, with registers preserved from one line to the next. In addition, there is a special set of debug commands. The set includes commands to display and change memory and/or registers, set and remove breakpoints (up to eight), locate strings in memory, display stack history and execute a specific number of instructions.

EMS 8000
Emulator System

Zilog

Product Description

September 1983

FEATURES

- Snapshot feature permits partitioning of a large real-time trace module into many small trace memories.
- Up to 126K bytes of high-speed, static, mappable memory can be accessed by the target system.
- Pulse output feature permits use of a high-end logic analyzer.
- Network debugging is supported.
- Full access to the target microprocessor's registers, memory, and I/O space is permitted.
- Transparent mode allows the same terminal to be used for host and EMS user interface.
- Emulates Z8001/3 or Z8002 CPUs at 6 MHz clock rates.
- Complex triggering.
- Large real-time trace buffer.
- Large mappable memory space.
- Real-time partitionable trace module for multiple recordings of program execution.
- Three parallel event comparators which can be allocated for trigger, trace, breakpoint recognition, and enable/disable functions.
- General-Purpose counter for benchmarking critical software routines.

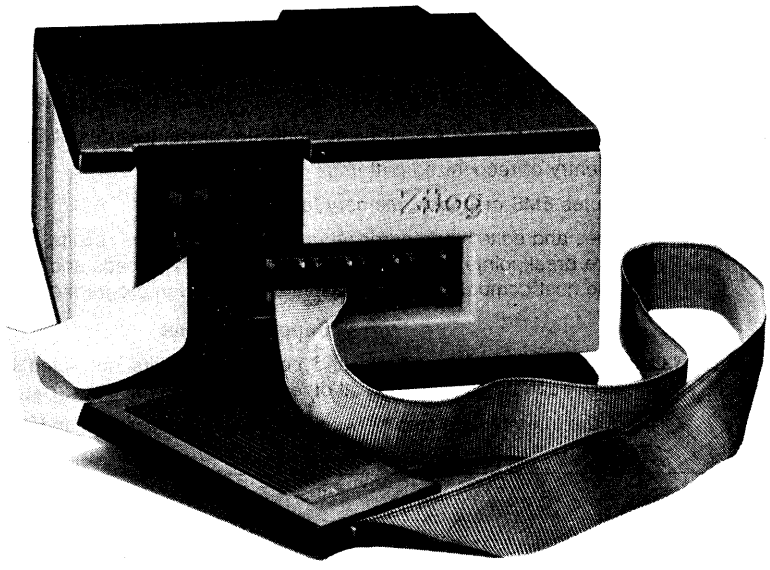


Figure 1. Emulator System 8000 (EMS)

GENERAL DESCRIPTION

The Emulator System 8000 (EMS), shown in self-test configuration in Figure 1, is a state-of-the-art in-circuit subsystem. The EMS supports the software/hardware engineer in developing products using the Zilog Z8000 family of microprocessors and peripheral components. Combined with Zilog's enhanced UNIX* System (ZEUS), EMS 8000 provides the designer with a complete and powerful set of tools for speeding up the product development cycle.

The EMS links the application software developed on a host system and the target system, and aids in the integration of the software into the target system by executing in a real-time environment. The EMS uses the full capability of the target microprocessor and can start or stop program execution or perform single-step execution. The user has full access to the target microprocessor's registers, memory, and I/O space.

The EMS is modular in design with a friendly, screen-oriented, self-prompting user interface. High performance is gained with a 1024 entry, real-time trace that is qualifiable and triggerable, and can be enabled with multilevel event recognition. Also, newly developed programs can be loaded into the development (target) hardware and executed in a real-time environment. The EMS can be networked into eight distinct Z8000 microprocessors that start and stop simultaneously.

Individual emulator systems can be defined as being either in or out of a break group. Systems which are outside of a break group can function as independent emulators with all of the EMS 8000 capabilities and full use of host resources. Systems which are in the break group are used to debug multiple processor systems.

SCREENS

The EMS is an interactive operating system that provides self-prompting commands and a set of powerful tools for complex debugging. The EMS command entry is organized into a set of pages called screens. Each screen is dedicated to a particular function and contains the commands and data fields necessary to accomplish specific functions. The screens are designed to fit on a standard display terminal, 80 columns wide by 24 lines long.

The user communicates with EMS through five command (menu-driven) screens and two support screens. The command screens are entered by typing the first letter of the screen name (e.g., A for allocation). Screens can be changed by sequentially typing <TAB> and the first letter of the screen name. Table 1 explains the screens and their functions.

Table 1. EMS Screen Descriptions

Screen	Function
	Command (Menu-Driven) Screens
Allocation	Assigns EMS resources to specific tasks such as tracing and breakpoints.
Configuration	Allows various hardware controls to configure global features of EMS.
Pattern	Allows entry of recognized patterns.
Mapping	Substitutes EMS mappable memory for target memory.
Debug	Examines and edits memory/registers, I/O, displays trace results, begins emulation, sets software breakpoints, turns watch area on and off, uploads and downloads files to and from the host computer, single/multiple steps through program execution.
	Support Screens
Help	Lists global command controls and helpful reminders not listed in the above menu screens. It can be displayed on all other screens by typing a <?>.
Change	This is the intermediate step between two command screens and is entered by typing a <tab>.

*UNIX is a trademark of Bell Laboratories.

HARDWARE DESCRIPTION

The EMS is a full-featured emulation peripheral. The heart of the EMS is a Central Controller Unit (CCU) with a 4 MHz Z80, 256K of dynamic memory, and 16K of ROM. The CCU contains the monitor program that provides a screen-oriented user interface, and operates continuously to allow the user to monitor the progress of emulation and breakpoints in real-time. The other EMS modules include a two-board Trigger module, a real-time Trace module, an External Probe interface module, a Mappable Memory module, and a microprocessor Personality module with a CPU Pod (Figure 2).

Figure 3 shows a fully configured EMS system with the following units:

- EMS 8000
- CPU Pod/cable assembly. The CPU Pod contains the processor chip to be emulated plus the required interface circuitry. Pods are available for the Z8001/3 and Z8002.
- 64K mappable memory (standard).
  62K mappable memory addition (optional).
- Host computer and user CRT terminal (required).
- External probes (optional).
- Target (the system being emulated).

The EMS uses dual-processor architecture to unburden the emulating processor from the configuration chores of the emulation system. (The Z80 CPU is used for EMS configuration and monitor functions and the Z8000 CPU for actual emulation.) This independence allows for improved debugging when unreliable target operation occurs.

A 10 MHz Z8000 CPU is used to emulate the 6 MHz maximum clock rate of the EMS to compensate for timing delays caused by buffering. The buffering provides better emulation control in problem targets and allows mappable memory to override existing target memory. Fast (90 ns) mapped memory allows emulation at 6 MHz with no Wait states. (Wait states can be forced if desired for compatibility with target memory.) Multilevel pattern recognition resources can be allocated in complex sequential, logical, and enable/disable combinations to the functions of trace qualifying and triggering, event counting, and timer modes. The counter/timer modes support a long count of 48 bits (40 when time is displayed in microseconds). This ensures adequate count capability for analysis of human-related events in real time.

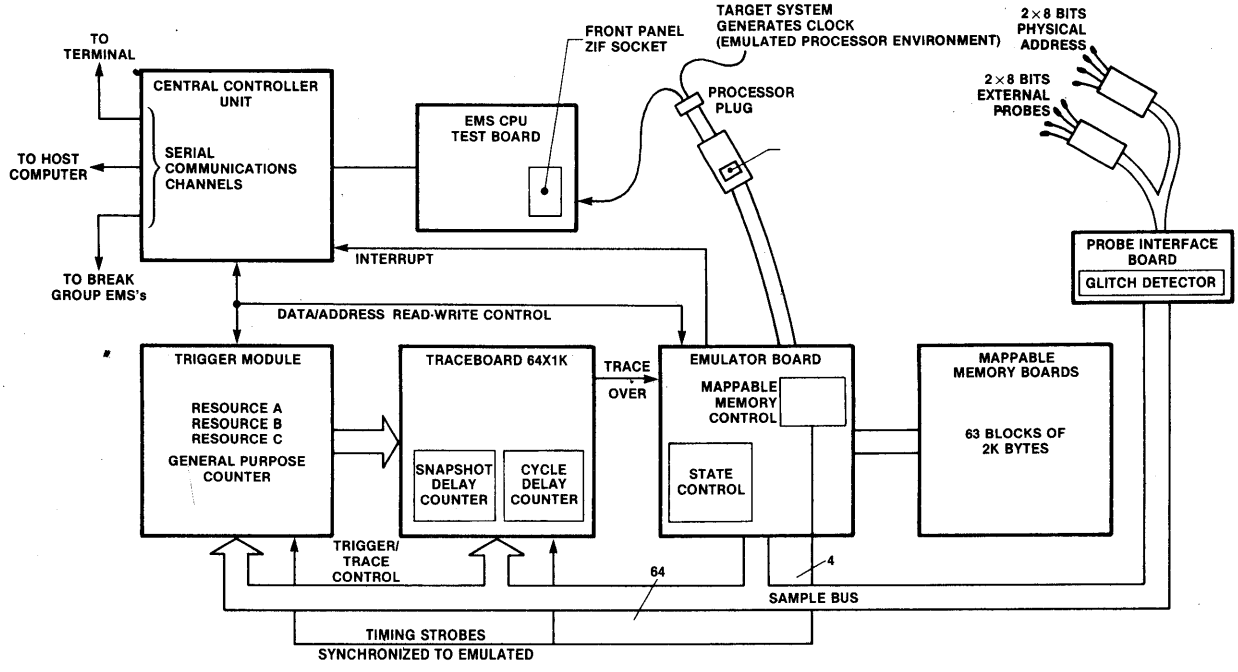


Figure 2. EMS 8000 System Block Diagram

SOFTWARE DESCRIPTION

The EMS can use ZEUS when the System 8000 is the host computer. This total system provides a complex hierarchical file structure that includes C, PLZ/SYS, a Z8000 assembler, a compiler writing system, and a general-purpose microprocessor. Because the EMS interfaces with Zilog computer systems, the user has access to powerful development tools for speeding up the product development cycle. Software downloads to either the ZEUS (UNIX) or RIO operating systems.

EMS software is friendly and easy to use. The menu prompt for each EMS screen reminds the user about the type of data that is available or the options that are permitted. Error checking prevents the user from entering illegal states and allows graceful recovery from emulation target problems (e.g., bad clock or power failure). Global command keys allow the user to control the starting and stopping of emulation, execution of command scripts, and entering Transparent mode independently of the command screens. A Help screen, which summarizes global commands and command entry, is available to help the user gain familiarity with EMS.

The EMS operating system is downloaded from a host computer, allowing easy implementation of future upgrades to improve its effectiveness and applicability. The hosts that can be configured with the EMS are:

- Zilog System 8000
- Vax UNIX
- PDP 11 UNIX
- Zilog MCZ 1, MCZ 2, and ZDS 1/40

The terminals that can be configured with EMS 8000 are:

- ADM 31
- CITOH
- Televideo 920
- VT 100
- VTZ 2/10

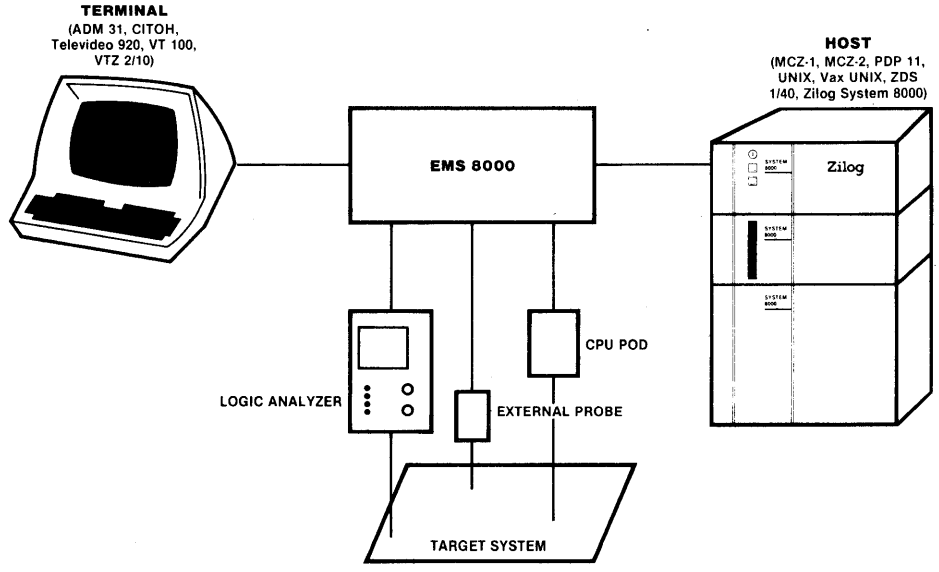


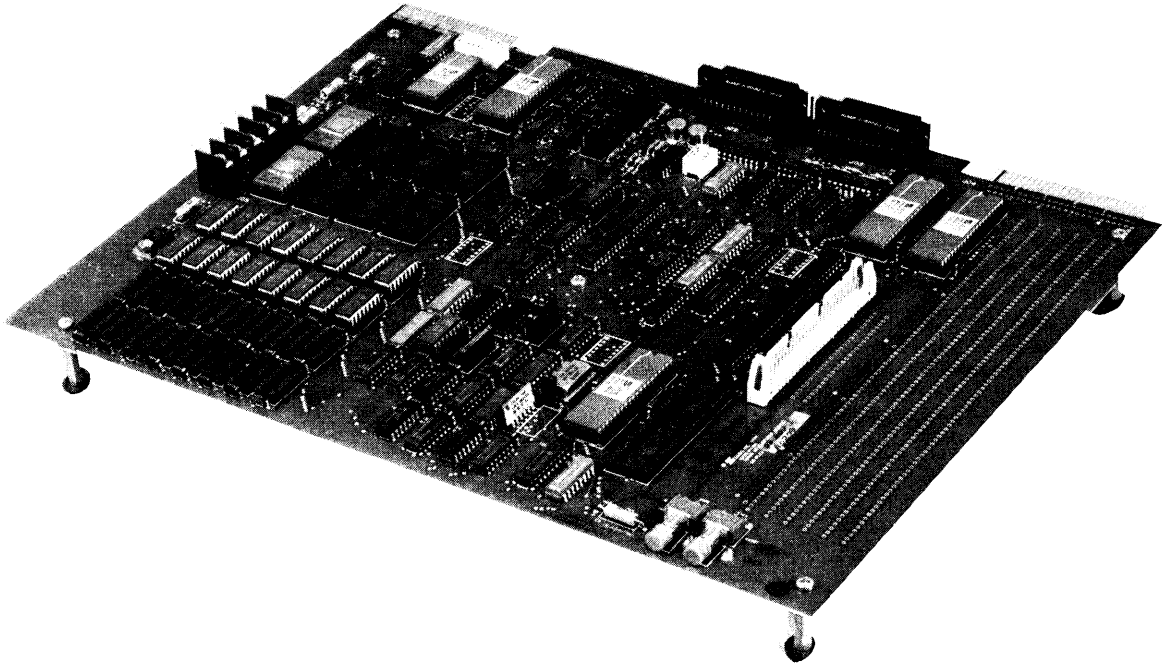
Figure 3. EMS 8000 System Configuration

Z8000™ Development Module

Zilog

Product Description

September 1983



- **Z8001/Z8002 CPU Evaluation and Debug Support**
- **16K Words Dynamic RAM (Expandable to 32K for User Code Execution and Debug**
- **32 Programmable I/O Lines**
- **EPROM Monitor and Debugger**
- **Transparent Operation Allows Software Development without Disconnection from CRT and Host System**
- **RS-232C Standard Serial Interfaces Compatible with Most CRT Terminals and Development Hosts**
- **Wire-wrap Area for Prototyping**

OVERVIEW

The Z8000 Development Module is a complete, single-board microcomputer that is used as a tool for the evaluation and debug of Z8000-based microprocessor systems. The Development Module is used in the first stages of the design and development process, not only as a tool for evaluating Z8000 microprocessor capabilities, but also as an environment in which code can be executed and debugged.

Evaluation. The Development Module provides a ready-made environment in which the user can execute software unique to his Z8000-based application, evaluate the CPU's performance, and then reach a realistic decision about its suitability for a specific application.

Software Debug. In addition to use as an evaluation tool, the Z8000 Development Module can be used to debug and modify user code. For the software designer, the Development Module is a real Z8000 environment in which he can execute code and carry out fairly extensive debugging. For the hardware designer, the Development Module is an example of Z8000 hardware design which provides special hooks and wire-wrap facilities to strap on additional logic.

FUNCTIONAL DESCRIPTION

Z8000 code developed on a software host may be downloaded serially to the Development Module RAM area via a serial port, and executed and debugged under EPROM monitor control. Once the system is connected, no further disconnection is necessary as the module has two serial ports (one connected to a host and the other connected to a CRT terminal). A simple software command makes the development process transparent in the serial path, thereby allowing direct communication between the host and terminal. The serial RS-232C interfaces allow virtually any software development host and CRT terminal to be used. For PROM-based code testing, the development module is self-contained and can operate stand-alone with a CRT terminal, since the host is only required for storage of user code on disk.

A variety of jumper areas and switches permit the selection of clock rates ranging from 2.5 to 3.9 MHz; the use of 2708, 2716, or 2732 EPROMs; the use of 4K or 16K RAMS; serial interface to modem, terminal, or teletype; I/O port addressing; and baud-rate selection from 110 to 19200 baud.

Hardware. The Z8000 Development Module is available in two versions: one supports the segmented Z8001 microprocessor; the other supports the non-segmented Z8002 microprocessor.

Z8001 Development Module. The Z8001 Development Module consists of a Z8001 CPU, 16K words of dynamic RAM (expandable to 32K words), 4K words of EPROM monitor (user-expandable to 8K words), a Z80A SIO providing dual serial ports, a Z80A CTC peripheral chip providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire-wrap area for prototyping hardware.

Z8002 Development Module. The Z8002 Development Module consists of a Z8002 CPU, 16K words of dynamic RAM (expandable to 24K words), 2K words of EPROM monitor (user-expandable to 8K words), a Z80A SIO device providing dual serial ports, a Z80A CTC peripheral device providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire-wrap area for prototyping.

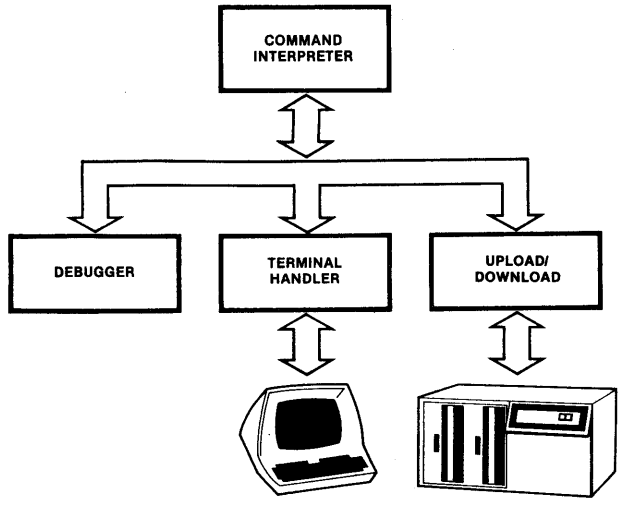


Figure 1. Monitor Block Diagram

Software. The monitor software (*Figure 1*) contained in EPROM (4K words for the Z8001 and 2K words for the Z8002) provides debugging commands, I/O control and host interface. It consists of a terminal handler, command interpreter, debugger and upload/download handler.

Terminal Handler. A Terminal Handler provides interface to the console device to facilitate output to a display or printing mechanism and input from a standard ASCII keyboard.

Debugger. The Debugger provides a basic set of debug commands to allow the user to start and stop program execution, display and alter CPU registers, flags or memory, and trap instruction sequences.

Command Interpreter. The Command Interpreter scans console inputs, ensures command validity and passes to other software modules in the monitor.

Upload/Download Handler. The Upload/Download Handler provides an interface between the serial connection and the host computer, the command interpreter and the memory resources of the Z8002 Development Module. It formats and interprets asynchronous data streams to and from the host and provides error checking and recovery for the serial interface (see *Figure 2*).

Memory Organization. *Tables 1 and 2* show the memory maps for the two versions of the Development Module. The organization of ROM and RAM in both the segmented and nonsegmented modes is indicated.

Figure 2. Serial Data Format

Address (Hex)	Memory
0000 0FFF	Monitor EPROM
1000 3FFF	User EPROM (User Installed)
4000 BFFF	Standard RAM
C000 FFFF	Expansion RAM *(User Installed)*

Table 1. Z8002 Development Module Memory Map

Segment 0		Segment 1	
Address (Hex)	**Memory**	**Address (Hex)**	**Memory**
0000 1FFF	Monitor EPROM	0000 3FFF	Expansion RAM (User Installed)
2000 3FFFF	User EPROM (User Installed)	4000 FFFF	Unused
4000 49FF	Monitor RAM (Scratchpad Area)		
4A00 BFFF	Standard RAM		
C000 FFFF	Expansion RAM (User Installed)		

Table 2. Z8001 Development Module Memory Map

MONITOR COMMAND SUMMARY

The following notation is used in the command description:

< > Enclose descriptive names for the quantities to be entered, and are not actually entered as part of the command.

[] Denote optional entries in the command syntax.

| Denotes "OR", eg. W|B denotes that either W or B may be used but not simultaneously.

< Prompt sign for the nonsegmented Z8002 monitor.

[Prompt sign for the segmented Z8001 monitor.

The following commands apply when the Z8001 monitor is used. All commands listed remain the same except those that permit reference to segmented addresses as follows:

<address> =
[<segment number>] <offset address>

<segment number> =
"<"<hex number in 7-bit range>">"

BREAK<address> [<n>] — Sets and clears a breakpoint at a given memory address. The option <n> allows specification of the number of occurrences, where n is from 1 to 128. The default is one.

COMPARE <address 1> <address 2><n> — Compares two blocks of memory data beginning with the addresses specified for <n> bytes, where n is from 1 to 128. Errors are reported on the console device.

DISPLAY<address> <n>[L|W|B] — Displays and modifies memory for <n> number of words or bytes. The optional entry allows data to be handled as bytes, words, or long words. The default is words.

FILL<address 1> <address 2><word> — Stores the <word> from memory address 1 to and including address 2.

GO — Begins program execution at the address contained in the current PC; execution is resumed where it was last interrupted. All registers are restored prior to execution.

IOPORT<address> [W|B] — Allows direct communications from the console to a selected I/O port. A word (W) or a byte (B) may be read from the selected port and a word or byte may be sent to the selected port; default is byte.

JUMP<address> — Unconditional branch to the specified address. All registers are restored prior to execution.

MOVE<address 1> <address 2><n> — Moves contents of a memory block from source address <address 1> to destination address <address 2> for <n> bytes.

NEXT[<n>] — Executes the next <n> machine instructions. <n> may be from 1 to 128. If n is omitted, 1 is assumed.

PUNCH<address 1> <address 2> — Punches a copy of memory from address 1 to address 2 on paper tape on the console device. Automatically turns on punch and a null leader is created. Upload/Download section describes the tape format used.

QUIT — Places serial channels into transparent mode. The Z8000 Development Module must be connected to both the Zilog host and the console device, and the Development Module acts as a message switcher.

REGISTER [<register name>] — Allows examination and modification of Z8000 registers. 8-bit, 16-bit or 32-bit quantities may be selected by the appropriate register-naming conventions.

TAPE — Loads memory from paper tape via the console device. The Upload/Download section describes the tape format used.

SPECIFICATIONS

Microprocessor
Z8001 or Z8002 CPU
Clock Rate: 2.5 MHZ or 3.9 MHz

Memory
ROM: 2K or 4K Words (Expandable to 8K Words)
RAM: 16K Words (Expandable to 32K Words)

Input/Output
Parallel: 32 Lines (Two Z80A-PIOs)
Serial: Dual RS-232C or RS-232C and Current Loop (Z80A-SIO)

Note
The user has access to all bus signals to allow custom system expansion into the wire-wrap area off-board.

Interrupts
Maskable Vectored (256), Maskable Non-vectored, Non-maskable, Segmentation Trap

Power
+5 V, 3 A
+12 V, 1 A
-12 V, 0.2 A

Physical

Height	1.75 in. (4.5 cm) Inclusive of Standoffs
Width	14.0 in. (35.6 cm)
Depth	11.0 in. (27.9 cm)
Weight	Approx. 30 oz. (850 gm)

ORDERING INFORMATION

Part No.	Description
05-6168-01	Z8001 Development Module
05-6101-01	Z8002 Development Module
05-6171-01	Z8001 Conversion Kit (converts Z8002 Development Module into Z8001 Development Module)

Systems recommended for use with the above:

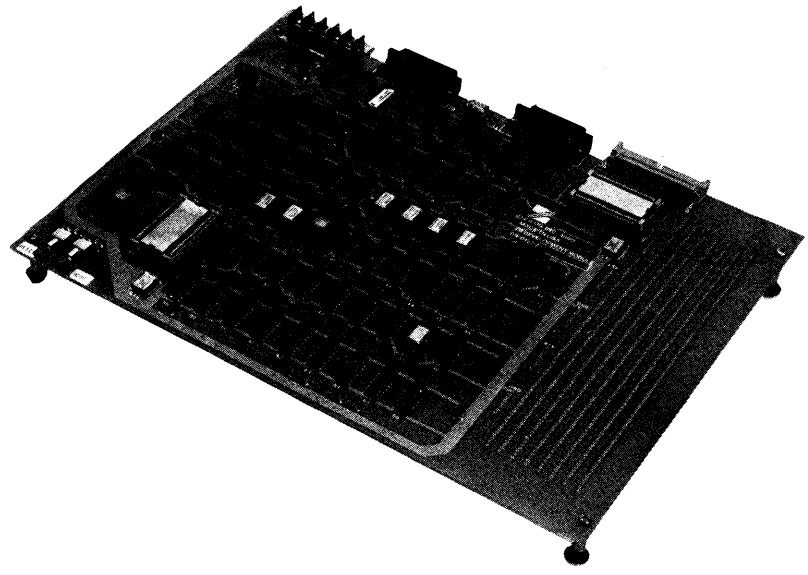
Description	Prerequisite
ZDS-1 Series Development Systems	Z8000 Software Development Package
PDS 8000 Series Development Systems	Z8000 Software Development Package
System 8000 Family	Z8000 Software Development Package

Z8® Development Module

Zilog NEW 1983 **Product Description**

September 1983

- Two 4K Z8s are used on the board: one as board manager and one for emulation (without real-time trace) or other user-defined configuration.
- 4096 bytes of static RAM allow convenient creation and debugging of user code.
- On-board socket tests user code in a 2716 or 2732 EPROM.
- Up to 4096 hardware breakpoints on address compare cover the entire internal ROM space.
- Versatile monitor software allows debugging, with register/memory examination and manipulation, and file upload and download.
- "Transparent" operation allows terminal-to-host communication without disconnecting the Development Module.
- Wire-wrap area for prototyping.
- Z8 board management is operated at 7.3728 MHz for baud rate purposes. The User Z8 has switch-selectable 8 or 12 MHz crystals.



OVERVIEW

The Z8 Development Module is a single-board microcomputer system specifically designed to assist in the development and evaluation of hardware and software designs based on the Z8 microcomputer family. It allows the user to build a prototype using the Z8 prototyping device, thereby developing code that will eventually be mask-programmed into the Z8 on-chip ROM.

Two Z8 devices exist on the Z8 Development Module: the Monitor Z8 serves as a board controller, while the User Z8 is user-definable. All user ports on the User Z8 are uncommitted and can be configured to suit any application.

Up to 4096 bytes of high-speed static RAM are available to simulate internal ROM. Also, an on-board EPROM socket allows the user to substitute EPROM for the ROM. This enables the user to store the software without building special hardware.

The EPROM-resident monitor software offers register and memory manipulation, as well as a convenient means to upload and download software between the host and user RAM space.

The Development Module connects to the CRT terminal and host system via two on-board RS-232-C serial ports; this places the Development Module between the CRT and host. A simple command makes the Development Module transparent in the serial path, which allows software to be developed on the host-resident assembler without disconnecting the Development Module from the CRT and host.

The Development Module can operate stand-alone for simple debugging operations, or it can interface directly to a host system such as the Zilog System 8000 for software development and file storage.

Fourteen square inches of wire-wrap area with 5 V and ground points are provided near the User Z8 for prototyping.

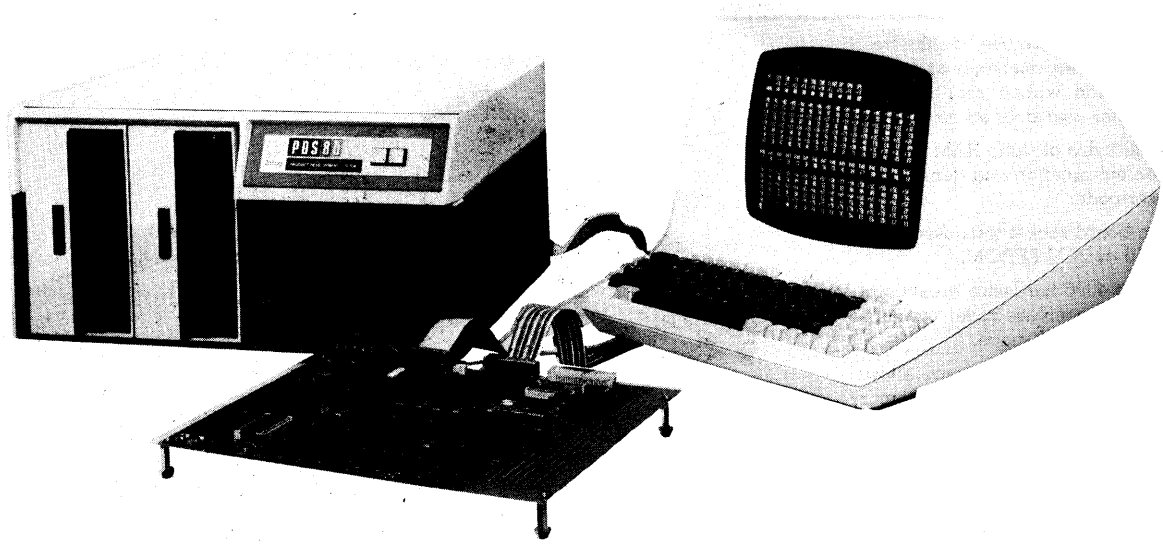
FUNCTIONAL DESCRIPTION

Hardware. The two Z8 microcomputer units (Monitor MCU and User MCU) are at the heart of the Z8 Development Module. The Monitor MCU controls operation of the User MCU using the monitor/debug software, which resides in 4K bytes of EPROM. Hardware breakpoint logic provides a maximum of 4096 breakpoints. Single-stepping with software trace capabilities is also available.

The User MCU is controlled by the Monitor MCU via internal address/data and control lines brought out to external pins. This effectively leaves all ports on the User MCU unconfigured and available for the user. The 4K bytes of static RAM on the internal bus are reserved for code that is executed by the User MCU. Execution is done in real time at full processor speed.

In addition to the wire-wrap area, a 40-pin header (3M type 2395-1002) for the User Z8 can connect to a ribbon cable with a 40-pin plug, which will then plug into a target system. Two switches, Mode and Reset, provide a means to re-enter the Monitor and to reinitialize the system, respectively. The baud rate, from 110 to 19200, is the same baud rate used for the terminal and host and is selected with an on-board, four-element DIP switch.

Software. The monitor/debug program includes debug, disassembly, input/output, control, and host interface commands. These commands are grouped into four major functional blocks: monitor, debug, manipulation, and file commands (see the following command list).



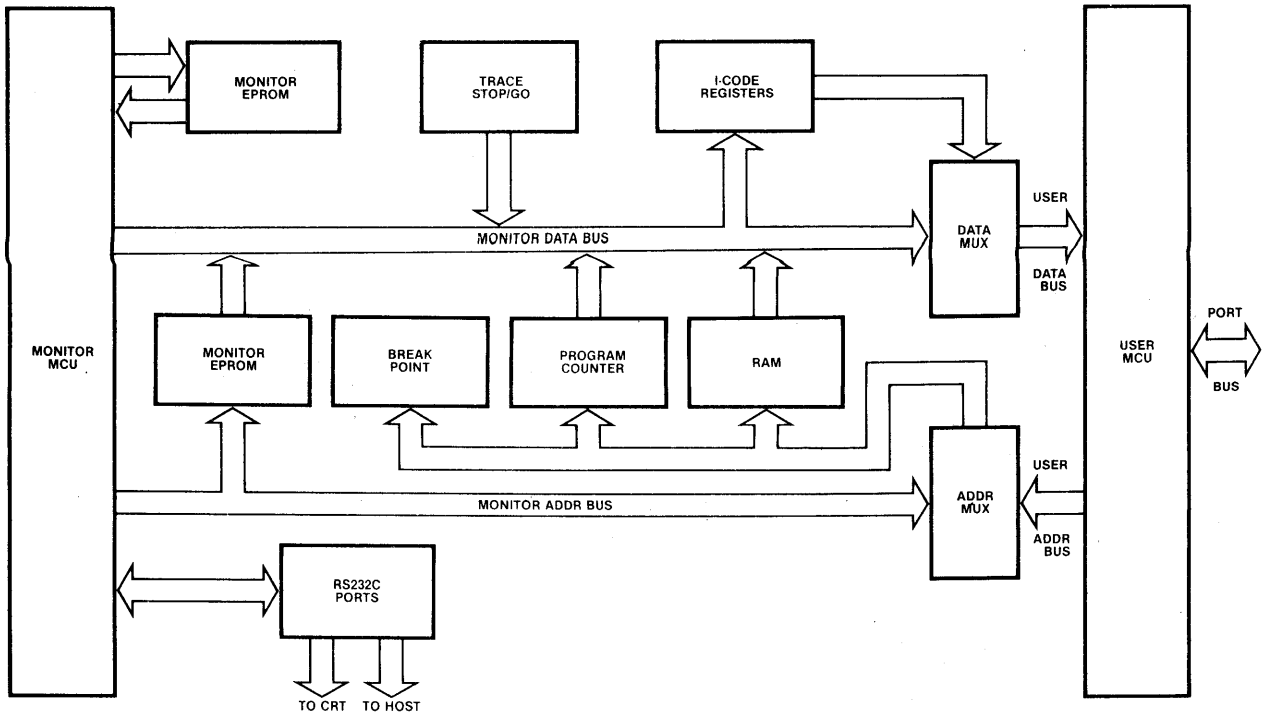
Z8 Development Module

Monitor Commands. This group of commands controls execution of the User MCU, monitors user interrupts and transfers control from the monitor to the host system.

GO <ADDRESS>	Causes User MCU to execute its program and disallows further debugging until a BREAK or HALT command is encountered.
HALT	Halts program execution of the User MCU.
QUIT	Returns control to the host system and enters the "transparent" mode.
INTERRUPTS [E/D]	Enables or disables all user-generated interrupts. Note: All user interrupts are automatically disabled when a breakpoint is encountered. It is necessary to reenable such interrupts with this command.

Debug Commands. This group of commands allows the user to debug code by tracing through code and setting breakpoints and jumps to specified locations within the "internal" ROM space, which is simulated in 4K bytes of RAM.

BREAK <ADDRESS>	Sets a breakpoint at the specified address.
KILL [<ADDRESS>]	Clears the breakpoint at the specified address.
JUMP <ADDRESS>	Allows the User MCU to jump to a specified address anywhere within the internal ROM space by changing the value of the Program Counter.
NEXT [<n>]	Causes execution of n instructions of the User MCU and then halts the User MCU.
TRACE	Causes single-step execution of the User MCU. Every instruction executed is output to the console.



Z8 Development Module Block Diagram

Manipulation Commands. The manipulation commands display and alter registers and memory. This group can be subdivided into two categories: register manipulation and memory manipulation.

Register Manipulation

REGISTER [<REG NUMBER>] [<NEW REG VALUE>]	Allows examination and modification of the Z8 internal registers.
WORKING REGISTERS	Displays contents of the current 16 working registers.
PHILL <STARTING REGISTER> <NUMBER OF REGISTERS> [<DATA BYTES>]	Stores the sequence of DATA BYTES into User MCU registers beginning at the STARTING REGISTER and continues for the NUMBER OF REGISTERS specified.

Memory Manipulation

DISPLAY [<STARTING ADDRESS>[<n>]]	Allows display and modification of user memory contents for n number of bytes.
SET <ADDRESS> <LENGTH> [<DATA BYTES>]	Allows a sequence of data bytes beginning at the ADDRESS specified to be written into user memory.
FILL <STARTING ADDRESS> <LENGTH> [<DATA BYTES>]	Stores the sequence of DATA BYTES into user memory beginning at the starting ADDRESS and continues for the LENGTH specified.
MOVE <SOURCE ADDRESS> <DESTINATION ADDRESS>[<n>]	Moves contents of a user memory block from a source address to a destination address for a length of n bytes.
COMPARE <ADDRESS 1> <ADDRESS 2>[<n>]	Compares two blocks of user memory data, one beginning at ADDRESS 1 and the other at ADDRESS 2 for n bytes.
ZAP[<STARTING ADDRESS>[<n>]]	Disassembles and displays code at a specified starting address for a specified number of bytes.

File Commands. The file group enables the user to upload and download programs to and from the host system.

LOAD <FILE NAME>	Downloads a file to user memory starting at the low address of the file and continuing until the entire file is transferred.
UPLOAD <FILE NAME> <ADDRESS 1> <NUMBER OF BYTES> [<ENTRY ADDRESS>]	Creates a RIO file image of user memory, beginning at ADDRESS 1, creating default length records, and imaging memory for the specified number of bytes.

Note: The following notation is used in the command description.

- < > Enclose descriptive names for the quantities to be entered, and are not actually entered as part of the command.
- [] Denote optional entries in the command syntax.
- | Denotes "or."

SPECIFICATIONS

Processor:
Two 64-pin DIP Z8s
Pin spacing is 0.070
Row spacing is 0.75

CPU Clock Frequency:
7.37 MHz for Monitor
8/12 MHz for User

Memory:
Monitor Z8
Scratch Pad RAM
RAM memory size: 1K bytes
RAM addressing: %2000 to %23FF
Minimum speed: 300 ns

EPROM
Word size: 8 bits
Memory size: 8K bytes
Addressing: 0 to %FFF internal
%1000 to %1FFF external
Minimum speed: 350 ns

User Z8
RAM (EPROM equivalent)
Word size: 8 bits
Memory size: 4K bytes
Addressing: 0 to %FFF (relative to User)
%9000 to %9FFF (relative to Monitor)
Minimum speed: 350 ns

Baud rate: Programmable to 110, 150, 300, 600, 1200, 2400, 4800, 19200 bps
Emulator cable length: 12 inches max.

Input/Output:
Monitor Z8
Baud rates: Programmable to 110, 150, 300, 600, 1200, 2400, 4800, 9600, 19200 bps
Connector type: Two 25-pin DB-25S connectors

User Z8
Parallel interface: 32 I/O lines undefined
Connector type: 40-pin PC edge connector

Dimensions (LxW):
29.94 cm (11 in.) x 35.56 cm (14½ in.)

Power Requirements:
1.4 A at +5 V dc ±5%

Environmental:
0 to 50°C (+32° to +122°F)
Up to 90% humidity without condensation

ORDERING INFORMATION

Part No.	Description
05-6158-01	Z8 Development Module (2K). Includes one serial interface ribbon cable and reference manual.
05-6222-01	Z8 Development Module (4K). Includes one serial interface ribbon cable and reference manual.

Systems recommended for use with above:

Description	Prerequisites
System 8000	Z8 Assembler

Z-SCAN UPC

Zilog NEW 1983 **Product Brief**

September 1983

FEATURES

- Portable, stand-alone, in-circuit emulator (ICE) for Zilog's Universal Peripheral Controller (UPC).
- Emulates six versions of the UPC: Z8093, Z8090, Z8094, Z8590, Z8593 and Z8594.
- Connects to the host and terminal via standard RS-232-C interface.
- Emulates Z-BUS and non-Z-BUS UPCs with either masked ROM or proto RAM/EPROM.
- Single-step trace capability.
- Monitor software allows file upload and download, register and memory manipulation.

OVERVIEW

The Z-SCAN UPC is a simple and cost-effective development tool that emulates four versions of Zilog's Universal Peripheral Controller (UPC). As a portable in-circuit emulator (ICE), the Z-SCAN UPC is an ideal tool for system development from design through manufacturing.

Both Z-BUS compatible and non-Z-BUS compatible types of UPC are emulated by the Z-SCAN UPC. The Z-BUS compatible Z-UPCs that are emulated are the Z8090 and Z8094. The non-Z-BUS compatible UPCs that are emulated are the Z8590 and the Z8594. Connection with the host and a terminal is accomplished via two RS-232-C interfaces.

By supporting eight popular terminal types and a wide variety of hosts, the Z-SCAN UPC is easily integrated into most operating environments.

FUNCTIONAL DESCRIPTION

The Z-SCAN UPC is physically located between the host system and the user's terminal, connected via the RS-232-C interface. The target cable connects directly to the front of the Z-SCAN for safety and convenience. The Z-SCAN UPC can operate in stand-alone mode for simple debugging operations, or it can be placed in transparent mode to allow software development with the host.

Hardware

The Z-SCAN UPC contains both a Z8 MCU and a UPC. The Z8 MCU controls monitor functioning, including operational commands and debug software. The UPC itself features a 256-byte register file including three I/O port registers, 234 general-purpose registers, and 19 control, status, and special I/O registers.

Twenty-four pins can be dedicated to I/O functions. These pins are grouped logically into three eight-line ports, which can be configured in various combinations as input or output, with or without handshake, and with push-pull or open-drain outputs.

Software

The monitor/debug program resides in 4096 bytes of PROM and contains debug, I/O, control, and host interface commands. This software is divided into four functional groups:

- Monitor commands control the Z8 MCU to monitor interrupts and transfer control from the monitor to the host system.
- Debug commands allow tracing and jumps to user-specified PROM locations.
- Manipulation commands permit display and alteration of registers and memory.
- File commands enable the user to upload and download to and from the host system.

In addition, Z-SCAN UPC software supports eight popular terminal types and easily interfaces to a wide variety of host systems.

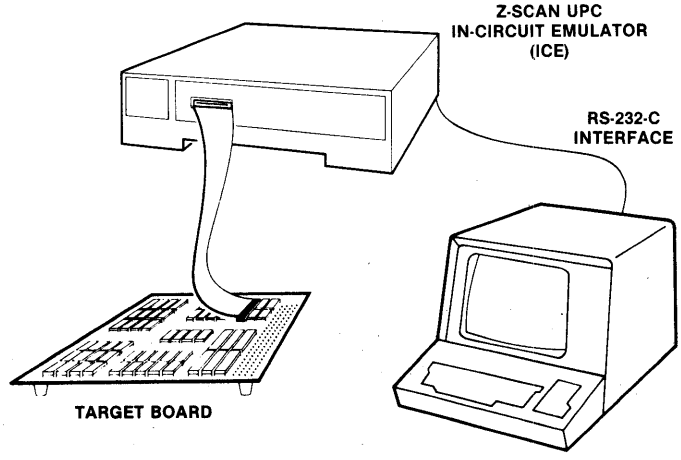


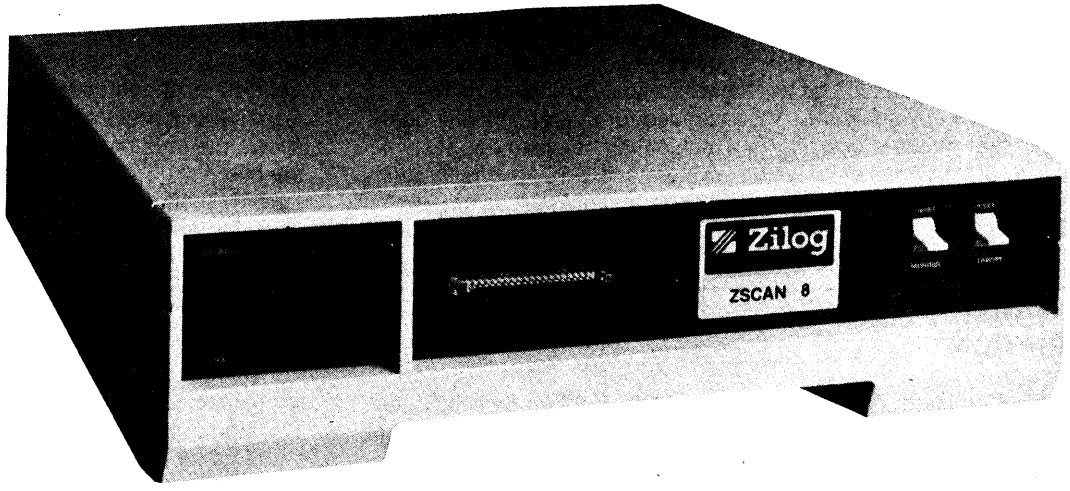
Figure 1. Stand-Alone Development System

Z-SCAN 8
Z8® Emulator

Zilog NEW 1983 **Product Description**

September 1983

- Provides real-time emulation capability for the family of Z8 Microcomputers.
- Operates with Zilog systems and other hosts; Z-SCAN 8 uses standard RS-232 links and is compatible with many standard CRTs and software hosts. This includes Zilog's S8000 systems, and others with user-supplied load/save routines and cross-software support.
- Hardware/software debugging is fast and convenient. Two screens display the status of the Z-SCAN 8 monitor and Z8 target resources. Target memory can be displayed and modified in a scrollable window.
- Fulfills the user's essential real-time debugging needs with its real-time trace, two complex breakpoints, single-step capability, and four blocks of mappable memory.
- Interactive and easy to use. Commands are selected from menus; command arguments are self-prompting.



OVERVIEW

The Z-SCAN 8 Emulator is a combination of hardware and software that allows efficient, interactive emulation of the Z8 Microcomputer. By the simple exchange of target devices, the selected Z8 MCU can be emulated in a realistic mode that allows user inspection and control over the environment being tested. Real-time trace, two breakpoints, single-step capability, and extensive mappable memory ensure the user a tool that accurately simulates the anticipated Z8 operating environment.

Z-SCAN 8 is an in-circuit emulator designed specifically for Zilog's Z8601 (2K), Z8611 (4K), and Z8681/82 (ROMless) Microcomputers. Z-SCAN 8 works with Zilog's family of development hosts, interfacing via two RS-232 serial ports to the host and a CRT terminal. A list of compatible CRT terminals is provided in Table 1.

Table 1. Terminals Supported by the Z-SCAN 8 Monitor

Manufacturer	Model
Lear Siegler	ADM31
Televideo	TVI 912 TVI 920
Zentec	Zephyr
Soroc	IQ 120 IQ 135
Beehive	Bee 100 Bee 107 Micro-B 1
DEC (any)	VT52 VT100 ANSI A3.64 or ISO DP 6429 compatible
General Terminals, Inc.	I-200 I-400
Hazeltine	1420 1500 Exec 80
Hewlett Packard	2620 2640
IBM	3101

Table 2. Recommended Sources for Cross-Software

Source	Description
Zilog 1315 Dell Avenue Campbell, CA 95008 (408) 370-8000	System 8000*. Cross-assembler for Z8 and Z8-UPC microcomputers.
Third Party	
Allen Ashley 395 Sierra Madre Villa Pasadena, CA 91107 (213) 793-5748	ASMB-Z8*. Cross-assembler; operates with any standard CP/M-based system. System-Z8*. Cross-assembler; includes ASMB-Z8 and text editor, operates with any standard CP/M-Z80-based system.
Avocet Systems, Inc. 804 South State St. Dover, DE 19901 (302) 734-0151	Z8 Cross-assembler (XASMZ8); operates with CP/M-80, CP/M-86, and MDOS.
Microtec P.O. Box 60337 Sunnyvale, CA 94088 (408) 733-2919	ASM Z8. Cross-assembler; operates with any general-purpose mainframe (DEC, IBM, DG, etc.) in FORTRAN.
Relational Memory Systems, Inc. P.O. Box 6719 San Jose, CA 95150	ASM Z8*. Relocatable macro cross-assembler; operates with Intel Intellec 800 and Series II microcomputer development systems.

*These include the upload and download software for communicating.

Because it uses a standard serial interface, Z-SCAN 8 can also be used with virtually any software host system that runs a cross-assembler or cross-compiler capable of generating Z8 code (see Table 2). This means software can be developed on many general-purpose computers. Only a simple upload and download utility is needed for operation since communication between the host system and Z-SCAN 8 is through a standard serial port using Tektronix hex format. Once software has been downloaded into the target, Z-SCAN 8 can be disconnected from the host and operated stand alone. Transparent operation allows the terminal to be used with the host in such a way that Z-SCAN 8 effectively disappears from the terminal-to-host link, without any physical re-cabling.

FUNCTIONAL DESCRIPTION

The Z-SCAN 8 emulator is a compact, portable device that can be used in a wide variety of functional configurations and applications. It has been designed to ease debugging of both hardware and software, to integrate hardware and software in Z8-based systems, and to provide the user with a powerful and versatile tool for the development of new systems and new applications for old systems.

The Z-SCAN 8 can be substituted for a Z8 microprocessor in any of its configurations or operational modes and can perform all the functions of the processor. Additionally, the Z-SCAN 8 allows the user to

- Control any function or operation of the processor and its internal (and in some cases, external) hardware.
- Inspect and display the condition or status of internal registers and CPU pins for up to 1,024 machine cycles preceding the breakpoint.
- Execute a program or any number of instructions in single step mode.
- Substitute up to 8K bytes of RAM for external program or data memory.

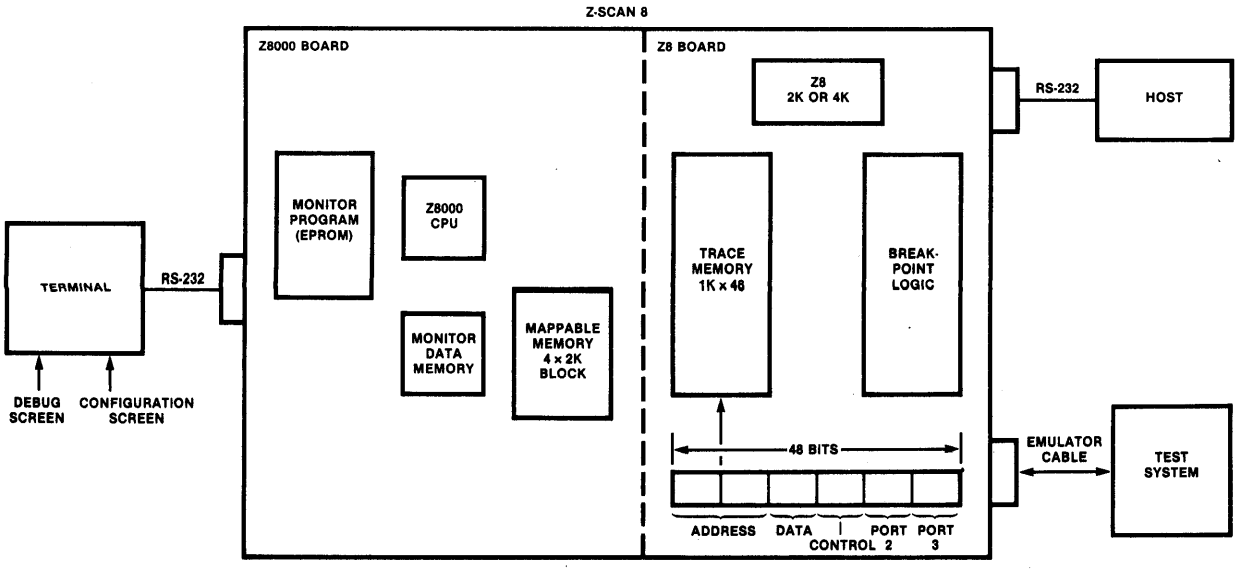


Figure 1. Z-SCAN 8 System Block Diagram

User Interface, Z-SCAN 8 Command Screens

All communication between the user and Z-SCAN 8 takes place through the terminal. The format consists of two selectable screen menus: the Configuration menu and the Debug menu. The operator can manipulate each of these primary menus to enable variations for which the user can select a particular set of conditions, such as parameters and other variables, for user control during emulation.

Z-SCAN 8 executes two types of commands: screen commands and manipulation commands. The latter are used to control the display on the monitor and are normally executed by a control character or an arrow key. Screen commands are those used to define and control the conditions of an emulation.

The Configuration Screen

The Configuration screen is primarily used to inform the Z-SCAN 8 of certain default values to be used for a specific emulation. It is comprised of five submenus:

- Host
- Load
- Save
- Map
- Target

These submenus allow the user to

- Set the host serial-link baud rate.
- Connect the user directly to the host system, in effect, making the Z-SCAN 8 transparent.
- Download programs or data from a host file.
- Allocate mappable memory in the Z-SCAN 8.
- Inform the Z-SCAN 8 of target configuration.

In practice, the Configuration screen is seldom changed after initial setup until some other type of test or exercise is contemplated.

The Debug Screen

Because it controls those conditions most often changed, the Debug screen is the screen most frequently entered during a series of tests or emulations. The Debug screen is comprised of five submenus:

- Watch
- Memory
- Break
- Xecute
- Display

The Watch command allows the user to designate up to twelve 16-byte lines of memory for display. This display is automat-

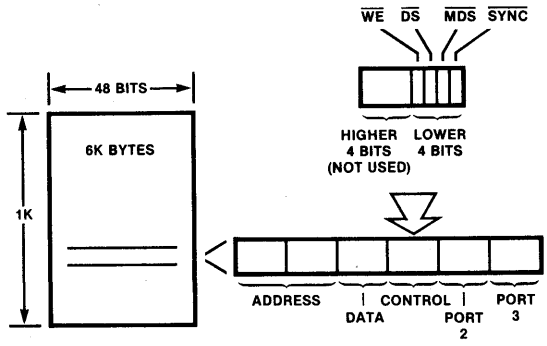


Figure 2. Bit Significance, Trace Memory Word

ically updated to show any change occurring in the section of memory specified.

The Memory command is used to compare two blocks of memory, fill a block of memory with a hexadecimal string, or move a block of memory to another range of addresses.

The Break command is used to define two complex breakpoints, which operate independently.

The Display command allows the user to specify what portion or range of program, data, or register memory is to be displayed on the screen.

Breakpoints

Two complex breakpoints are available in the Z-SCAN 8. Each breakpoint can be programmed independently to stop all processor activity at some arbitrarily selected point and save the state of the system in the trace memory for later analysis. The breakpoints may specify a stop on address, on data, or on a status such as an interrupt acknowledge. Or the breakpoint may specify that a pulse be generated and sent to the BNC connector at the back of the machine rather than stopping the emulation.

Trace Memory

The trace memory of the Z-SCAN 8 consists of a 48-bit by 1K block of RAM that can be used by the operator to record the condition and status of certain elements of the processor's environment for up to 1,024 machine cycles. The trace memory can then be displayed and the display used to analyze an entire series of steps in a routine. The bit significance of the 48-bit trace memory word is shown in Figure 2.

Mappable Memory

Mappable memory in the Z-SCAN 8 consists of four 2K blocks of high-speed static RAM. Each of the blocks can be assigned independently to replace a section or block of the target system's memory. The block can be assigned anywhere in the Z8's memory space and can be specified to respond to program or data memory or both. Mapping must be done on 2K word boundaries only, and the entire block can be write-protected. When a break results from a write-protect violation, an error message appears on the CRT.

SOFTWARE DESCRIPTION

The basic design of the Z-SCAN 8 software divides the emulator functions into two main tasks, the Net Task and the Command Task. The third major program module is the Kernel, which performs the functions of an operating system.

The two system tasks are written in the C programming language. To take advantage of the system of pointers available in this language, the system software has made extensive use of tables that define the collection of system commands.

INTERFACE TO NON-ZILOG HOSTS

Load/save communication between a Zilog (or other) host system and the Z-SCAN 8 monitor is accomplished by exchanging messages containing printable ASCII characters. Message types are:

- Single-character, data-block acknowledgment
- Error text
- Data block

All messages exchanged during a Load or Save command are text lines, each ending in RETURN (carriage return). Memory and other data are converted into hexadecimal numerals for transmission, and the resultant message is readable left-to-right, high-order digit first, as it is transmitted over the RS-232 link.

Z-SCAN 8 SPECIFICATIONS

Processor:
40-pin, 2K and 4K Z8 CPU

Clock Rate:
Internal to Z8000 board: 2.4 MHz

Emulation Frequency:
Up to 12 MHz

I/O:
Two RS-232-C serial ports for terminal and host

CRT Terminal:
Any standard CRT system, including Zilog's S8000 systems.

Baud Rate:
Terminal:
9600
Host:
Determined by user selection from 300 to 38,000

Mappable Memory:
8K high-speed, static RAM assignable in 2K blocks

Breakpoints:
Two comples breakpoints; breakable on data, address, or interrupt acknowledge

Emulator Cable:
24 inches

Front Panel:
TARGET RESET and MONITOR RESET switches, POWER ON indicator, 40-pin connector type.

Rear Panel:
BNC connector for pulse output, standard LS-TTL level 2 x 25 pin connectors, 3M type 3483 (terminal and host), 3-pin power connector, 1¼ in., fuseholder (screwdriver release type), POWER ON switch rocker type), 115/220 voltage selection switch (sliding type)

Power:
180-264 volts ac or 90-130 volts ac, switch selectable; 47-63 Hz; 2 amp maximum

Dimensions:
4 inches x 17.5 inches x 14.5 inches (HWD); 10.2 centimeters x 44.5 centimeters x 36.8 centimeters

Environment:
10°C to 50°C (operating)

Unit Weight:
25 pounds

ORDERING INFORMATION

Part No.	Description
05-0144-00	Z-SCAN 8 Emulator

Z-SCAN 800
Z800™ Emulator

Zilog NEW 1984

Product Description

September 1983

INTRODUCTION

The Z-SCAN 800 is a low-cost Z800™ MPU development system designed to facilitate the integrated hardware and software development of Z800-based systems. The Z-SCAN 800 represents the logical choice for Z800 development, and offers Z80® CPU system developers the ideal window to the future.

Based on Zilog's 16-bit Z8000™ CPU and connected via two RS-232-C compatible ports, the Z-SCAN 800 (Figure 1) offers in-circuit emulation (ICE) in a stand-alone mode or in a hosted multi-user configuration, using any of several common terminals and various host development systems.

The stand-alone mode is especially useful in manufacturing environments as it provides simple testing and debugging for PROM-based target systems. Configured with a host, the Z-SCAN 800 becomes transparent, lending itself to the development of high-level software.

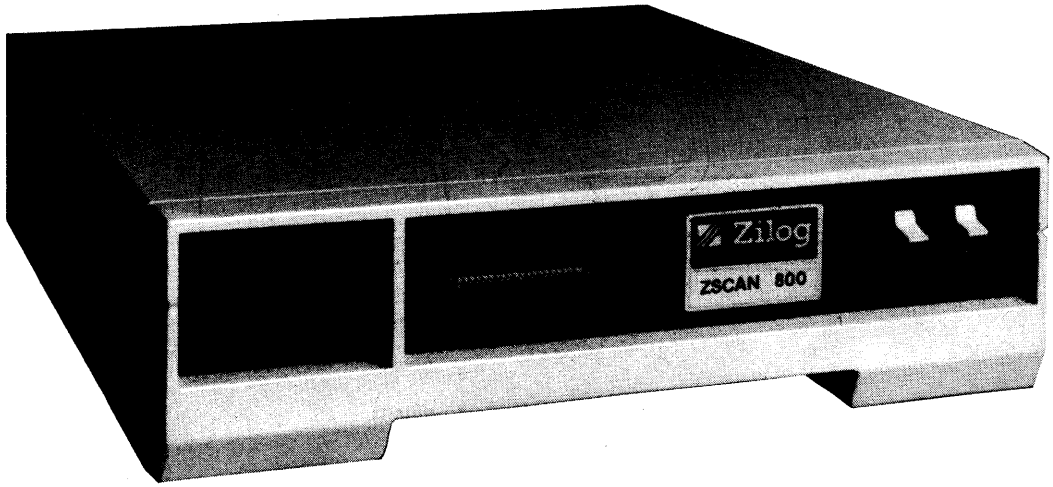


Figure 1. Z-SCAN 800

Z-SCAN software offers similar flexibility. The user at a terminal has a choice of screens to view, and each screen offers a variety of options to enable the user to focus on specific needs. Real-time emulation, extensive trace capability, user-defined breakpoints, and a large mappable memory help maintain the Zilog tradition of advanced, friendly development systems. Figure 2 illustrates the Z-SCAN 800 environment.

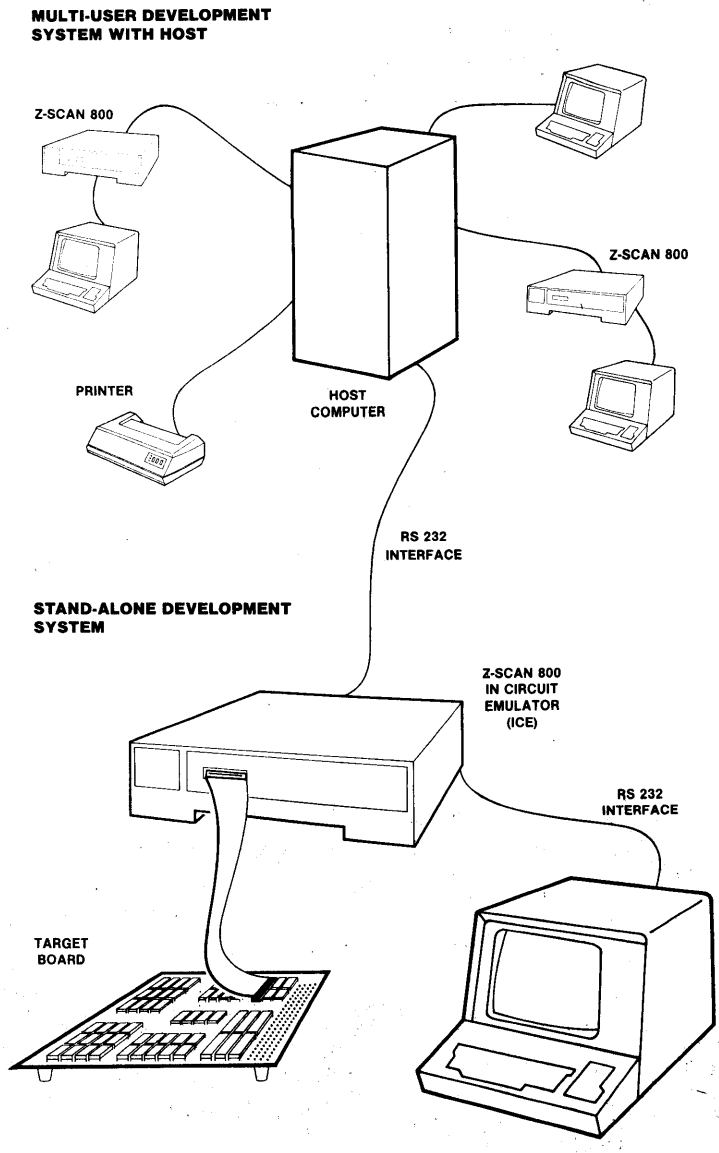


Figure 2. Z-SCAN 800 Environment

Z800 EMULATION

The Z-SCAN 800 emulates all four versions of the Z800 MPU. These four versions are the Z8108, Z8208, Z8116, and Z8216. All Z800s are binary-code compatible with the Z80, and incorporate advanced architectural features such as dual mode (User/System) operation, on-chip peripherals, and memory management.

The Z8108/8208 MPUs are characterized by an 8-bit, Z80-type bus. The Z8108, a 40-pin package, includes on-chip the Z800-standard Memory Management Unit (MMU), clock oscillator, and refresh controller. The Z8208, a 64-pin package, additionally includes four 24-bit Direct Memory Access (DMA) channels, four 16-bit counter/timers, and a full-duplex UART to facilitate asynchronous serial communication.

The Z8116/8216 MPUs operate on the 16-bit Z-BUS® . As with the Z8108/8208 CPUs, the Z8116 contains the advanced features standard to the Z800s and the Z8216 includes the additional on-chip peripherals.

Regardless of the package type, the Z-SCAN 800 traces all address, data, status, and control signals on Z800 pins for development with any Z800 MPU.

MAPPABLE MEMORY

In addition to providing MPU emulation during system development, the Z-SCAN 800 also emulates user-board ROM by providing 64K bytes of static RAM. This RAM may be divided in up to eight blocks of 8K bytes each and is mappable on any 4K byte boundary. A special write-protection feature can be individually applied to any combination of the eight blocks, providing a safe and accurate representation of system ROM.

The system developer can use the mappable memory as a complete substitute for eventual on-board ROM, or as a partial supplement to existing ROM. This supplement may represent additional ROM under development or even "throw away" ROM to be used only during debugging.

SCREEN-ORIENTED DISPLAY

The screen-oriented display is one example of the user-friendly software of the Z-SCAN 800. The screen-oriented approach frees the user from the need to remember all the possible commands because the options are listed, and it eliminates time-consuming and error-producing keystrokes by requiring only cursor placement for selection. A first-time user can effectively learn the Z-SCAN 800 in a few hours by interacting with the terminal display. The user chooses between two screens: the Debug Screen and the Configuration Screen.

Debug Screen

The Debug screen acts as a "window" into the target. The user can configure the screen to display preformatted presentations of target resources such as special-purpose and general-purpose registers; target memory can be displayed in scrollable fashion.

Configuration Screen

The Configuration screen provides multiple options regarding the set-up of mappable memory. This screen contains information less frequently used during the debugging phase, such as baud rates and target characteristics.

INTERFACE PROTOCOLS

Z-SCAN 800 is provided with protocols to permit uploading and downloading with many host computer systems. Protocols are provided for the Z800 family running under several operating systems as shown in Table 1. These standard protocols allow system developers to use their current hardware configuration as a host for Z-SCAN development work. In addition, protocols are provided to allow Z-SCAN 800 to interface with such popular terminals as ADM-31s, Hazeltine, HP and many more.

Table 1. Z-SCAN 800 Interface Protocols

Host Operating System	Z800 Mode Protocol
Apple	Special
CP/M	ELOAD-binary
Intel MDS	ELOAD-binary
System 8000, ZEUS	ELOAD-binary
VAX, UNIX*	ELOAD-binary

*UNIX is a registered trademark of Western Electric

BREAKPOINTS

The Z-SCAN 800 provides two complex hardware breakpoints and fifteen software breakpoints. The two hardware breakpoints permit the user to break on address, data, and control signals, and can cause breaks on specific combinations of conditions. The additional software breakpoints contribute to making debugging a relatively simple task.

Additionally, the Z-SCAN 800 incorporates a break-in/break-out feature, extremely useful in a Local Area Network (LAN) debugging situation. Break-in and break-out pulses allow simultaneous breaking within a multiple Z-SCAN environment. The break-in signal may come from an external source or it may originate in a previously set hardware breakpoint within a Z-SCAN 800.

TRACE

Another invaluable tool for debugging is the trace feature. Trace records up to 64-bits of data on up to 1000 events, creating clear histories of Z800 signal states.

Finally, a disassembler provides the user with easily recognizable mnemonics to interpret results.

And the Z-SCAN 800 is an exceptional value. It comes at a very low cost and includes a pod to ensure reliable results even with high frequencies and long cables. The Z-SCAN 800 is fully UL/CSA approved.

*UNIX is a trademark of Bell Laboratories. Zilog is licensed by AT&T.

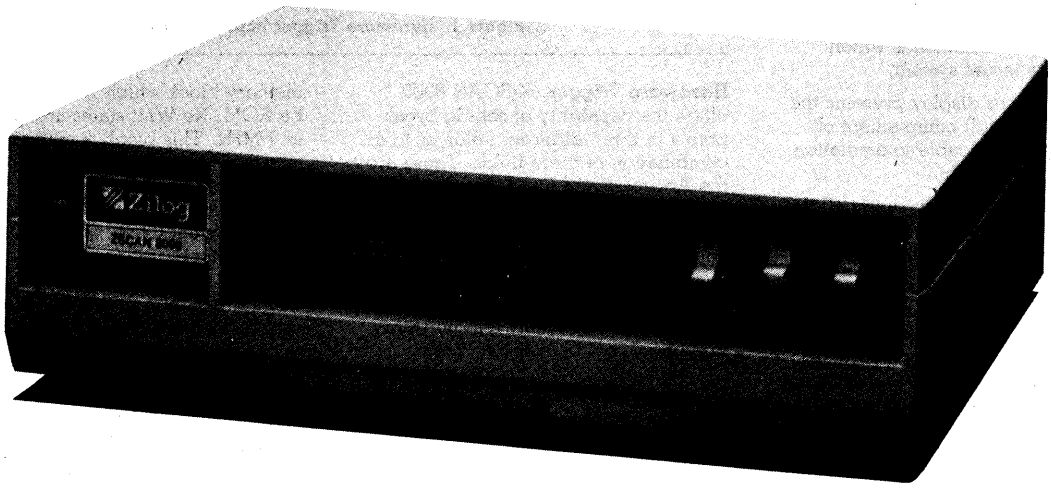
Z8000™ Emulator
Z-SCAN 8000

Zilog

Product Description

September 1983

- **Provides Real Time Emulation up to 4 MHz of the Z8001 and Z8002 CPUs.**
- **Two RS-232C Serial Ports Make It a Peripheral Usable with Most Standard CRTs and Software Hosts.**
- **Transparent Operation Permits Direct Communication Between CRT and Host without Physical Disconnect.**
- **Highly Interactive, Screen-Oriented User Interface Makes Z-SCAN Easy To Use.**
- **Shadow Monitor Removes All Restrictions on Target System Memory Space, Making It Fully Available To the User.**
- **High-Speed Mappable Memory (no wait states) Is Available to Simulate Target System RAM/ROM.**



OVERVIEW

The Z-SCAN 8000 Emulator is an in-circuit emulator that has been designed as a peripheral unit for Zilog's Z8001 and Z8002 16-bit microprocessors. Interfacing via two RS-232C Serial ports to host and CRT terminal, Z-SCAN 8000 can work with Zilog's family of development hosts.

Because it employs a standard serial interface, Z-SCAN 8000 can also be used with virtually any software host system that runs a cross assembler or cross compiler capable of generating Z8000 code. Communication between the host system and Z-SCAN 8000 is with a standard serial format requiring only a simple upload and download utility to operate. For PROM-based target systems, Z-SCAN can operate stand-alone with a CRT terminal because the monitor and debug software is EPROM-resident.

In keeping with Zilog's design philosophy of separating a development system into two identifiable units (the software host and an emulation peripheral), Z-SCAN 8000 fits into three scenarios, making it a highly versatile unit:

- As a peripheral to Zilog's PDS 8000, ZDS-1, or System 8000, Z-SCAN 8000 completes the development support package for the Z8001 and Z8002 microprocessors available from Zilog.
- As a peripheral to any development host with the capability of compiling or assembling Z8000 code, Z-SCAN 8000 allows a low-cost emulation capability which precludes substantial reinvestment in a software host system.
- As a stand-alone in-circuit emulator that can operate with most CRT terminals, Z-SCAN 8000 provides simple testing and debugging capability for PROM-based target systems.

SYSTEM FEATURES

User Interface. Z-SCAN 8000 incorporates the use of a two-dimensional screen-oriented user interface which makes it easy to use. Because it is general-purpose in nature, the user interface does not require a customized CRT terminal to operate. The only requirements are that the CRT terminal have screen erase, line erase, and cursor addressing capability.

The objective of the user interface is to provide a screen format with a menu-like approach, which directs the user through the operation of the emulator. The user is aware at all times of where he/she is in the debug process because Z-SCAN 8000 provides the CRT information about system parameters, system resources, current execution, and error messages. When the system is turned on, a bootstrap routine produces a display informing the user of the unit's configuration and requesting the user to define set-up parameters. A menu of display choices shows the user the different capabilities of the system:

- The Memory/I/O command display shows the various memory and I/O manipulation commands which access the target system.
- The Resources display presents the user with the full complement of arguments applicable to emulation of the target system.
- The Execution display shows all the commands and parameters necessary to cause emulation to take place.

At all times, execution of specific Monitor commands is possible, and information on other relevant system parameters and resources is always displayed. This highly interactive user interface makes it possible to use Z-SCAN 8000 without frequent reference to the operating manual.

Shadow Memory. Z-SCAN 8000 is a single, CPU-based system that can be configured to emulate either the Z8001 or Z8002 by simply exchanging the CPU, monitor EPROM, and the emulator cable.

Although the system uses a single CPU for both monitor and emulation functions, no restrictions are placed on the target system memory size. This is because the entire monitor resides in shadow memory and, therefore, does not appear in the target system memory space. This feature also provides the benefit of making future system expansion possible without any hardware redesign.

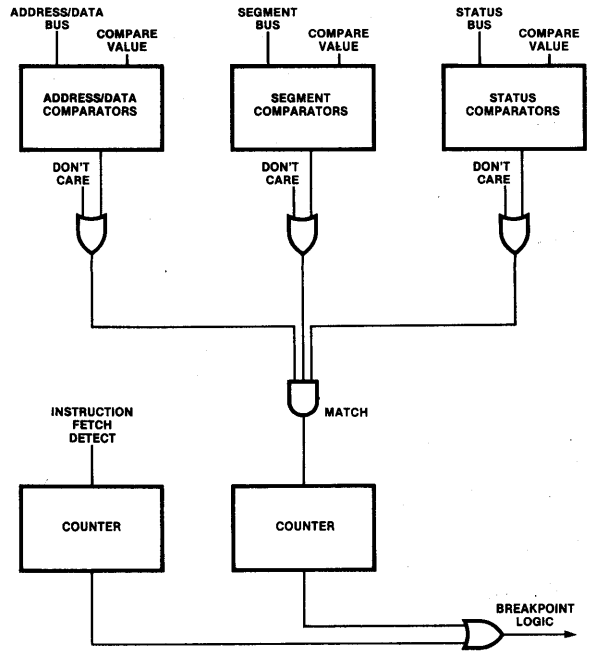


Figure 1. Hardware Trigger Implementation

Hardware Trigger. Z-SCAN 8000 offers the capability of setting breakpoints in three different fields or in a combination of these fields. These are the Address/Data Field, the Segment Field, and the Control/Status Field. A Pass Counter can be set up to a maximum of 255 counts to allow multiple pass triggering. In addition, Z-SCAN 8000 may also be set to break on instruction fetches only (single-step execution), or, by using a Pass Counter, may be set up to a maximum of 247 counts to allow triggering on multiple instruction fetches (multi-step execution).

With these two capabilities, a breakpoint argument can be set up which is on ORed condition allowing for either a break-on-field (or combination of fields) argument or for "n" instruction fetches, whichever occurs first. This ORed situation is convenient when tracing through a program in search of a specific occurrence. A pulse output, providing a trigger pulse on breakpoint match condition is available on the rear panel to trigger auxiliary test instrumentation.

Mappable Memory. Z-SCAN 8000 offers a 4K work block of high-speed static RAM. This block is available to the user to simulate a target system memory block which would typically be ROM. No Wait states are required at 4 MHz. This block is mappable anywhere in the Z8001 and Z8002 address space and can be specified to be Normal Code, Normal Data, Normal Stack, System Code, System Data, System Stack, or Space Independent. Mapping must be done on 4K word boundaries only, and the entire block can be write protected against illegal writes to cause system emulation either to break on such occurrences or continue emulation. An error message appears on the CRT display informing the user of an illegal write.

Software Trace. Z-SCAN 8000 offers a software trace feature which provides insight into target system activity and CPU resources. In the Trace Mode, the system displays the address of the instruction being executed and the contents of the CPU registers (both general-purpose and control) consecutively, covering one full screen format.

For example, displaying the CPU registers associated with every instruction executed just prior to executing a Break is tremendously useful to the user during debug of target system activity.

SPECIFICATIONS

CPU
Z8001 or Z8002 per configuration

Clock Rate
500 kHz–4.0 MHz (external)

I/O
Two RS-232C Serial Ports for CRT and host

Baud Rate
Automatically selected from 50 to 19.2K

Breakpoint
Address, Data, Segment and Address, Control, Address and Control, Data and Control, Segment and Address and Control, Instruction Fetch, OR combination of Instruction Fetch and any Field argument

Mappable Memory
4096 × 16 Static RAM (no Wait states at 4 MHz while operating off User clock)

Inputs
One standard LS-TTL load plus 30 pF maximum

Outputs
Capable of driving one standard LS-TTL load plus 30 pF preload

Rear Panel Output
BNC connector for pulse output, standard LS-TTL

Front Panel
Target/Monitor, Reset, and NMI toggle switches

Power
110/220 Vac, 50/60 Hz switch selectable, 60 VA maximum

Dimensions
4 in. (10.2 cm) (H) × 14½ in. (36.8 cm) (W) × 18 in. (45.7 cm) (D)

Emulator Cable
12 inches

AC CHARACTERISTICS

Number	Symbol	Parameter	Z8001/2 Min (ns)	Z8001/2 Max (ns)	Z-SCAN Min (ns)	Z-SCAN Max (ns)
1	TcC	Clock Cycle Time	250	2000	250	2000
2	TwCh	Clock Width (High)	105	2000	105	2000
3	TwCl	Clock Width (Low)	105	2000	105	2000
4	TfC	Clock Fall Time		20		20
5	TrC	Clock Rise Time		20		20
6	TdC(SNv)	Clock ↑ to Segment Number Valid (50 pF load)		130		175
7	TdC(SNn)	Clock ↑ to Segment Number Not Valid	20		35	
8	TdC(Bz)	Clock ↑ to Bus Float		65		165
9	TdC(A)	Clock ↑ to Address Valid		100		163
10	TdC(Az)	Clock ↑ to Address Float		65		154
11	TdA(DI)	Address Valid to Data In Required Valid	455		383	
12	TsDI(C)	Data In to Clock ↓ Setup Time	50		76	
13	TdDS(A)	$\overline{DS}$ ↑ to Address Active	80		-4	
14	TdC(DO)	Clock ↑ to Data Out Valid		100		163
15	ThDI(DS)	Data In to $\overline{DS}$ ↑ Hold Time	0		-20	
16	TdDO(DS)	Data Out Valid to $\overline{DS}$ ↑ Delay	295		269	
17	TdA(MR)	Address Valid to $\overline{MREQ}$ ↓ Delay	55		29	
18	TdC(MR)	Clock ↓ to MREQ ↓ Delay		80		143
19a	TwMRh	$\overline{MREQ}$ Width (High)	210		193	
19b	TwMRh	$\overline{MREQ}$ Width (High) During Monitor Operation			184	
20	TdMR(A)	$\overline{MREQ}$ ↓ to Address Not Active	70		53	
21	TdDO(DSW)	Data Out Valid to $\overline{DS}$ ↓ (Write) Delay	55		59	
22	TdMR(DI)	$\overline{MREQ}$ ↓ to Data In Required Valid	350		287	
23	TdC(MR)	Clock ↓ $\overline{MREQ}$ ↑ Delay		80		134
24	TdC(ASf)	Clock ↑ to $\overline{AS}$ ↓ Delay		80		134
25	TdA(AS)	Address Valid to $\overline{AS}$ ↑ Delay	55		29	
26	TdC(ASr)	Clock ↓ to $\overline{AS}$ ↑ Delay		90		144
27	TdAS(DI)	$\overline{AS}$ ↑ to Data In Required Valid	340		277	
28	TdDS(AS)	$\overline{DS}$ ↑ to $\overline{AS}$ ↓ Delay	70		53	
29	TwAS	$\overline{AS}$ Width (Low)	70		53	
30	TdAS(A)	$\overline{AS}$ ↑ to Address Not Active Delay	60		43	
31	TdAz(DSR)	Address Float to $\overline{DS}$ (Read) ↓ Delay	0		-41	4

CONTINUED ON NEXT PAGE

AC CHARACTERISTICS

Number	Symbol	Parameter	Z8001/2 Min (ns)	Z8001/2 Max (ns)	Z-SCAN Min (ns)	Z-SCAN Max (ns)
32	TdAS(DSR)	$\overline{AS}$ ↑ to $\overline{DS}$ (Read) ↓ Delay	70		53	
33	TdDSR(DI)	$\overline{DS}$ (Read) ↓ to Data In Required Valid	185		122	
34	TdC(DSr)	Clock ↓ to $\overline{DS}$ ↑ Delay		70		65
35	TdDS(DO)	$\overline{DS}$ ↑ to Data Out and STATUS Not Valid	75		58	
36	TdA(DSR)	Address Valid to $\overline{DS}$ (Read) ↓ Delay	180		154	
37	TdC(DSR)	Clock ↑ to $\overline{DS}$ (Read) ↓ Delay		120		174
38	TwDSR	$\overline{DS}$ (Read) Width (Low)	275		258	
39	TdC(DSW)	Clock ↓ to $\overline{DS}$ (Write) ↓ Delay		95		149
40	TwDSW	$\overline{DS}$ (Write) Width (Low)	185		168	
41	TdDSI(DI)	$\overline{DS}$ (Input) ↓ to Data In Required Valid	320		266	
42	TdC(DSf)	Clock ↓ to $\overline{DS}$ (I/O) ↓ Delay		120		174
43	TwDS	$\overline{DS}$ (I/O) Width (Low)	410		393	
44	TdAS(DSA)	$\overline{AS}$ ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay	1065		1048	
45	TdC(DSA)	Clock ↑ to $\overline{DS}$ (Acknowledge) ↓ Delay		120		174
46	TdDSA(DI)	$\overline{DS}$ (Acknowledge) ↓ to Data In Required Delay	435		381	
47	TdC(S)	Clock ↑ to Status Valid Delay		110		162
48	TdS(AS)	Status Valid to $\overline{AS}$ ↑ Delay	60		45	
49	TsR(C)	$\overline{RESET}$ to Clock ↑ Setup Time	180		208	
50	ThR(C)	$\overline{RESET}$ to Clock ↑ Hold Time	0		15	
51	TwNMI	$\overline{NMI}$ Width (Low)	100		116	
52	TsNMI(C)	$\overline{NMI}$ to Clock ↑ Setup Time	140		154	
53	TsVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Setup Time	110		118	
54	ThVI(C)	$\overline{VI}$, $\overline{NVI}$ to Clock ↑ Hold Time	0		22	
55	TsSGT(C)	$\overline{SEGT}$ to Clock ↑ Setup Time	70		78	
56	ThSGT(C)	$\overline{SEGT}$ to Clock ↑ Hold Time	0		22	
57	TsMI(C)	$\overline{MI}$ to Clock ↑ Setup Time	180		188	
58	ThMI(C)	$\overline{MI}$ to Clock ↑ Hold Time	0		22	
59	TdC(MO)	Clock ↑ to $\overline{MO}$ Delay		120		165
60	TsSTP(C)	$\overline{STOP}$ to Clock ↓ Setup Time	140		148	
61	ThSTP(C)	$\overline{STOP}$ to Clock ↓ Hold Time	0		22	
62	TsWT(C)	$\overline{WAIT}$ to Clock ↓ Setup Time	50		78	
63	ThWT(C)	$\overline{WAIT}$ to Clock ↓ Hold Time	10		25	
64	TsBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Setup Time	90		98	
65	ThBRQ(C)	$\overline{BUSREQ}$ to Clock ↑ Hold Time	10		32	
66	TdC(BAKr)	Clock ↑ to $\overline{BUSACK}$ ↑ Delay		100		145
67	TdC(BAKf)	Clock ↑ to $\overline{BUSACK}$ ↓ Delay		100		145

ORDERING INFORMATION

Part No.	Description
05-0100-00	Z-SCAN 8000/1 Emulator (Supports Z8001 Emulation and Control)
05-0100-01	Z-SCAN 8000/2 Emulator (Supports Z8002 Emulation and Control)
05-0101-00	Z8001 Field Support Kit (Converts Z-SCAN 8000/2 into Z-SCAN 8000/1)
05-0102-00	Z8002 Field Support Kit (Converts Z-SCAN 8000/1 into Z-SCAN 8000/2)
05-0103-01	Z-SCAN 8000 Emulator Includes Z8001 and Z8002 CPU's, emulator cables and serial interface cables.

Systems recommended:

Description	Prerequisites
ZDS-1 Series Development Systems	Z8000 SDP
PDS 8000 Series Development Systems	Z8000 SDP
System 8000 Family	Z8000 SDP

System 8000
Models 21 and 31

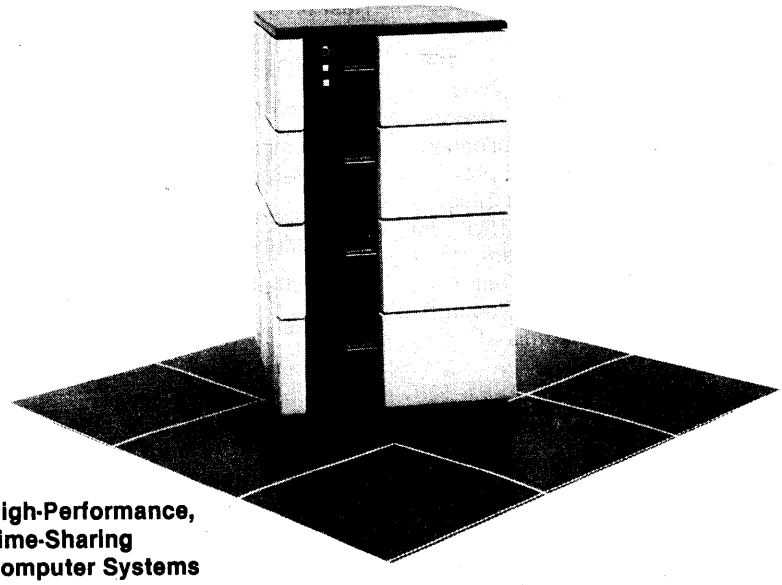
Zilog

Product Brief

September 1983

Features

- VLSI components include the 6 MHz Z8001A CPU and 6 MHz Z8010A Memory Management Units (MMUs), as well as 6 MHz Z80B CPUs. The Z8000 is a 16-bit CPU that has 16 general-purpose registers, an 8M byte address space, and capability to perform 8-bit, 16-bit, and 32-bit operations.
- ZEUS, Zilog's enhanced UNIX* operating system, is a complete, real UNIX, not a look-alike. Enhancements include additional utilities from Zilog and the University of California at Berkeley.
- System 8000 uses a powerful screen-oriented text editor to increase editing speed.
- A selection of high-level languages is offered for the Z8000: BASIC, COBOL, C, FORTRAN 77, Pascal and PLZ/SYS.
- Up to 4M bytes of ECC-controlled memory increases system reliability, and a large physical memory size improves system performance.
- Up to 24 users can be supported.
- Optional industry-standard 9-track, half-inch, magnetic tape is available for information interchange. Streamer mode is used for disk backup.



High-Performance, Time-Sharing Computer Systems

The System 8000 Models 21 and 31 are general-purpose, multiuser, time-sharing computer systems. They offer versatile and economic solutions for the commercial user, since many users can be added to the same system, all performing different tasks simultaneously. Sharing of files and transmission of messages between users is easy. The multiuser programming environment allows both sophisticated and untrained users to operate the computer.

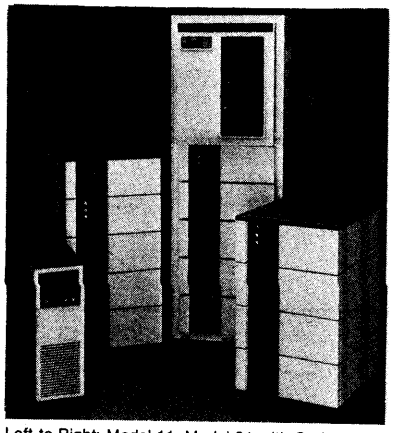
State-of-the-Art Technology Plus Future Expansion

For optimal performance, the System 8000 utilizes many advanced components such as Z8000™ CPU, ECC (Error Checking and Correction controlled memory, Winchester disks, cartridge tape and low power consumption devices. System growth is made possible by the addition of more boards and more peripheral modules.

As a supplier of both components and systems, Zilog incorporates the latest microprocessor technology into system-level products.

The System 8000 family offers high-performance systems at reasonable cost.

*UNIX is a trademark of Bell Laboratories. Zilog is licensed by Western Electric, Inc. for the Seventh Editon and System III.



Left to Right: Model 11, Model 31 with Optional Expansion Chassis. Model 31 with Optional 9-Track Tape Drive. Model 21.

Software

For the operating system, Zilog chose to implement UNIX, the optimum environment for application software development and programmer productivity. The Zilog "port" of UNIX is named ZEUS, for Zilog Enhanced Unix System. It contains many additional features and utilities designed by Zilog and the University of California at Berkeley.

Unlike some other implementations of the popular UNIX operating system, ZEUS, in conjunction with the Z8001 CPU and the specialized memory management hardware in the System 8000, is able to support extremely large user programs. This can be especially important for large FORTRAN and C applications.

ZEUS is a general-purpose, multiuser, multitasking operating system designed for software development. Productivity increases with ZEUS because with it, the necessary software tools are available. Documentation quality and availability also improve because tedious jobs are automated.

All of the software needed to make system programmers, application programmers, and technical writers more productive is available on the system. Commercial users can choose from COBOL and BASIC business languages; system programmers have C, Z8000 assembler, and PLZ/SYS for their needs. Technical users can select FORTRAN 77 or Pascal.

Programs for business such as Accounts Receivable, Payroll, and Order Entry can be quickly ported and customized. In the technical environment, special software and test programs can be developed to fit the needs of the task. The major operating system features are:

- Hierarchical file system
- Compatible file, device, and interprocess input/output
- Separate code and data address space
- Multiple processes per user
- User configurability
- User programs address space of up to 8M bytes (C, FORTRAN 77, assembler)

System Utilities

The system utilities include the command interpreter and file maintenance, status inquiry, system accounting programs, and data communications. The command interpreter is selected on a per-user basis, enabling the system to be tailored to the needs of different users. Data communications utilities are included for networking over a serial link to other local or remote ZEUS- or UNIX-based computer systems.

The ZEUS programming tools consist of languages, libraries, a symbolic debugger, text processing software, and more than 180 other utilities.

Hardware

The System 8000 hardware is designed to support ZEUS software. The memory management architecture allows ZEUS to support, with minimal changes, programs that run under the UNIX operating system. The memory architecture also makes it possible for user programs to have an address space of up to 8M bytes.

Designed for performance, reliability, and future growth, System 8000 is based on Zilog's 6 MHz Z8001A CPU, with high-performance, high-reliability Winchester and Storage Module Disks (SMD). Intelligent I/O controllers aid performance. The large ECC-controlled memory reduces swapping. The Z-BUS Backplane Interconnect (ZBI™) and modular system packaging allow for system growth. Serial RS-232-C and parallel interfaces allow attachment of I/O devices such as CRTs and printers.

The modular design makes System 8000 easy to service. As many as five modules can be stacked. Each module is self-contained so that modules can be unstacked without tools. The unit can also be mounted in a standard 19″ rack.

Zilog is committed to the long-term development of the System 8000 product family. A recent addition is an industry-standard, half-inch magnetic tape peripheral option. This peripheral supports interchange of information on tape with other systems and can also be used in a "streaming" mode for fast, high-capacity backup of the on-line disks.

System 8000 Model 21 and 31 Characteristics

Physical

Height	84 cm (33 in.)
Width	48 cm (19 in.)
Depth	61 cm (24 in.)
Weight	105 kg. (250 pounds) approximate

Winchester Disk Performance

Rotation Speed	3,600 RPM
Power ON to Ready Time	60 seconds
Average Positioning Time (Typical)	45 ms
Number of Surfaces	4
Tracks per Surface	600
Sectors per Track	24
Bytes per Sector	512
Data Transfer Rate	801K bytes/sec
Capacity Unformatted	32M bytes

Storage Module Disk (SMD) Performance

Rotation Speed	3,600 RPM
Power ON to Ready Time	60 seconds
Average Positioning Time (Typical)	20 ms
Number of Surfaces	7
Tracks per Surface	589
Sectors per Track	32
Bytes per Sector	512
Data Transfer Rate	1.2M bytes/sec
Capacity Unformatted	84M bytes

Environmental

Operating Temperature	10° C (50° F) min. 40° C (104° F) max.
Relative Humidity	20-80% (noncondensing)

Cartridge Tape Drive Performance

Speed Read/Write (rewind/search)	30 ips (90 ips)
Tracks	4
Recording Density	6,400 BPI

Memory

1M bytes ECC (minimum)

Electrical

Phase	Single
Frequency	47-63 Hz

Nominal Selectable Voltages (±10%)	Current Sustained (max)	Current Surge (max)
100 V ac/60 Hz	5.0 A	8.0 A
117 V ac/60 Hz	5.0 A	8.0 A
220 V ac/50 Hz	2.5 A	4.0 A
240 V ac/50 Hz	2.5 A	4.0 A

System 8000 Model 11

Zilog

Product Brief

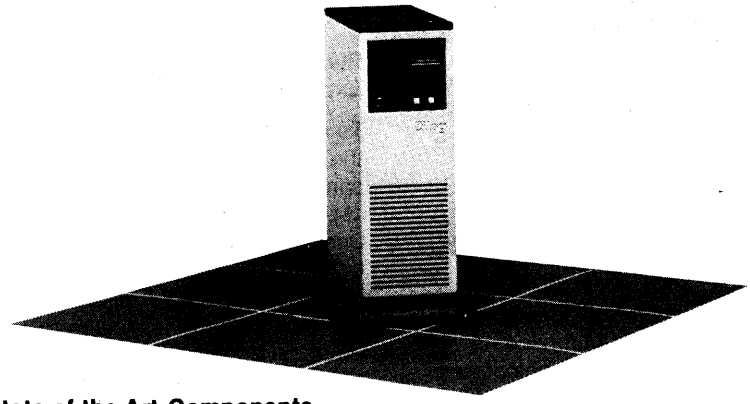
September 1983

Features

- Economical, time-sharing system supports up to eight users.
- Powerful ZEUS operating system, Zilog's implementation of UNIX*.
- ZEUS includes a screen-oriented text editor and comprehensive text processing software for fast and easy editing applications.
- Model 11 supports such high-level languages as COBOL, BASIC, C, PLZ/SYS, FORTRAN 77, and Pascal.
- Up to 1M bytes of parity memory or optional 1M byte of ECC-controlled memory.
- Up to 36M bytes of disk storage capacity.
- 17M byte cartridge tape for backup and archiving.

A Compact Computer Powerful Enough to Support Eight Users

The Zilog System 8000 Model 11 has been specifically designed for business and office environments. The system is small enough to fit under a desk, yet can provide computing power for up to eight users simultaneously. This general-purpose, time-sharing system is ideally suited for commercial needs.



State-of-the-Art Components

The Model 11 features reliable, field-proven VLSI components such as the Z8001A CPU (which supports separate code and data address spaces) and three Z8010A Memory Management Units.

The basic Model 11 has 256K bytes of parity random-access memory (RAM). A 5¼-inch Winchester disk provides 18M bytes of storage. The sealed, high-performance Winchester disk drive protects data from contaminants and provides economical, fast, and highly reliable on-line storage of data.

A 17M byte cartridge tape drive is used for hard disk backup. This tape drive offers virtually unlimited off-line storage of data.

Powerful ZEUS Operating System

For the operating system, Zilog chose to implement UNIX, the optimum environment for application software development and programmer productivity. The Zilog "port" of UNIX is named ZEUS, for Zilog Enhanced Unix System.

Unlike some other implementations of the popular UNIX operating system, ZEUS, in conjunction with the Z8001 CPU and the specialized memory management hardware in the System 8000, is able to support extremely large user programs. This can be especially important for large FORTRAN and C applications.

*UNIX is a trademark of Bell Laboratories. Zilog is licensed by Western Electric, Inc. for the Seventh Edition and System III.

ZEUS includes features and additional software developed by Zilog and the University of California at Berkeley. Major features of the ZEUS operating system are its hierarchical file structure and compatible file, device, and interprocess input/output functions.

ZEUS includes an impressive array of programs that comprise its system utilities and development tools. These software tools make it easy for both OEMs and end-users to develop new applications quickly.

Reliable Hardware

The system's hardware has been designed to complement its powerful software. Also, programs developed on other units in the System 8000 family are completely transferrable to the Model 11.

Intelligent Winchester disk and tape drive controllers free the operating system from routine device-handling functions.

The system's 32-bit Z-BUS Backplane Interconnect (ZBI™) provides flexibility in system configuration and allows for future expansion.

Serial RS-232-C and parallel interfaces allow the system to communicate with input/output devices such as CRTs and printers.

Flexible Hardware Expansion

To accommodate system growth, the basic Model 11 System 8000 can be expanded to include up to 1M bytes of parity or ECC- (Error Checking and Correction) controlled memory. A second 5¼-inch, 18M byte Winchester disk drive can be added to the system to provide dual drives with a total of 36M bytes of on-line data storage capacity.

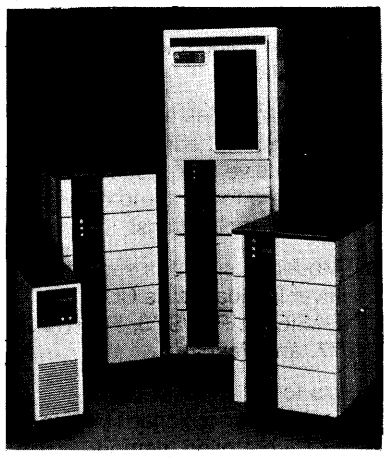
System Utilities

The ZEUS system utilities provide programs for user access, command processing, file management, status information, communication with other devices or systems, and system maintenance.

ZEUS features a command interpreter as an interface between the user and the system. It is selected on a per-user basis, allowing the system to be tailored to meet the needs of different users.

The ZEUS development tools include programming languages, libraries, a symbolic debugger, text processing and formatting software, and more than 180 other utilities. ZEUS also includes a two-dimensional, screen-oriented text editor to increase editing speed.

The system supports several high-level programming languages for business, scientific, and industrial applications. Choose from COBOL, BASIC, C, PLZ/SYS, FORTRAN 77, and Pascal.



Left to Right: Model 11, Model 31 with Optional Expansion Chassis. Model 31 with Optional 9-Track Tape Drive. Model 21.

System 8000 Model 11 Characteristics

Physical

Height	66 cm (26 in.)
Width	20 cm (8 in.)
Depth	46 cm (18 in.)
Weight	43 kg. (95 pounds) approximate

Winchester Disk Performance

Rotation Speed	3,600 RPM
Power ON to Ready Time	60 seconds
Average Positioning Time (Typical)	75 ms
Bytes per Sector	512
Data Transfer Rate	5M bits/sec
Capacity Unformatted	18M bytes

Environmental

Operating Temperature	10° C (50° F) min. 40° C (104° F) max.
Relative Humidity	20-80% (noncondensing)

Cartridge Tape Drive Performance

Speed Read/Write (rewind/search)	30 ips (90 ips)
Tracks	4
Recording Density	6,400 BPI

Memory

256K-byte parity (minimum)

Electrical

Phase	Single
Frequency	47-63 Hz

Nominal Selectable Voltages (± 10%)	Current Sustained (max)	Current Surge (max)
100 V ac/60 Hz	2.5 A	4.0 A
117 V ac/60 Hz	2.5 A	4.0 A
220 V ac/50 Hz	1.25 A	2.0 A
240 V ac/50 Hz	1.25 A	2.0 A

Software Development Products

Zilog

Pioneering the Microworld

Development Systems Software for All Zilog Microprocessors

Zilog offers full support for all its microprocessors and microcomputers with the System 8000. The System 8000 computer system uses ZEUS, the UNIX*-based operating system specifically designed for software development and text processing. Numerous development tools are available, including the programming languages PLZ/SYS, C, FORTRAN 77, Ada, and Pascal; various libraries; and a symbolic debugger.

Cross-software packages run under ZEUS on the S8000 enable complete code development for all Zilog microprocessors and microcomputers. This includes C compilers for the Z8000 and Z80 and assemblers for the Z8000, Z800, Z80, and Z8.

Emulators designed for the complete range of Zilog products are detailed in the Development Section of this data book. Because ZEUS treats emulators as System 8000 peripherals, System 8000 can be combined with EMS 8000, Z-SCAN 8000, or with non-Zilog emulators to provide total product development support for multiple microprocessors.

The ZRTS Kernel is a small executive program that provides the core of a real-time multitasking operating system in PROMable form. In addition to development languages, ZRTS 8000 and ZCL are available to assist the user in implementing a real-time target operating system. ZCL is a high-level configuration language that permits the designer to configure the target system in easy-to-use statements.

More software products for Zilog components and systems are described in the Zilog Software Catalog.

*UNIX is a trademark of Bell Laboratories.

Z8000™ PLZ/SYS

Zilog

Product Brief

June 1982

Features

- High-level procedure-oriented language permits efficient writing of machine-independent modules and programs.
- Structured format for fast and easy-to-compile programs.
- Produces efficient code for economical memory usage and processing time.
- Simplifies software production and maintenance.
- Allows direct or interpretive execution of program modules.
- Supports both segmented and nonsegmented Z8000 processors.

Description

Z8000 PLZ is a family of different programming languages designed to satisfy a wide range of microcomputer software development requirements. The two members of the PLZ family, PLZ/SYS and PLZ/ASM, produce object code-compatible modules and share common control structures and data definition facilities. Thus, selective portions of programs can be written in the most appropriate language for the specific application and still maintain a consistent structure between modules.

PLZ/SYS is a high-level, procedure-oriented language that is syntactically similar to Pascal. It provides a medium for writing structured, machine-independent programs with a minimum of programming effort.

PLZ/ASM, on the other hand, is a structured assembly language that permits access to the low-level capabilities of the processor by mixing assembly language and high-level control structures.

Compiler. The Z8000 PLZ/SYS Compiler translates source code modules into an intermediate stage called Z-code. The Z-code modules can then be executed interpretively or processed by the code generator to produce a machine-code object module.

The compiler provides support for both the segmented and non-segmented Z8000 processors.

Code Generator. The Z8000 PLZCG Code Generator accepts a file of intermediate Z-code generated by PLZ/SYS and produces the corresponding Z8000 machine code as a relocatable object module. This file can be linked with other modules to form a complete executable load module.

Interpreter. The intermediate Z-code modules produced by the Z8000 PLZ/SYS Compiler can be executed interpretively by ZINTERP. Linking ZINTERP with the other modules generated by the compiler produces an executable load module.

Linker. The Linker, ZLINK, links Z-code, ZINTERP and/or machine code modules into a single relocatable load module, allowing the user to control the overall size and speed of the program.

Although interpretive Z-code runs more slowly than machine code, the space savings over machine code is usually substantial for larger programs where the 3K bytes of ZINTERP is a small percentage of the entire program. By balancing the number of Z-code and machine code modules, the user can maximize the efficiency of a particular program.

ZLINK resolves any external references between separately assembled modules, so that the load module produced is relocatable. It also allows the reordering and combining of named sections between modules and supports incremental linking.

Operating Environment. Z8000 PLZ/SYS is supported on Zilog's S8000 Development System.

Z8070 Floating-Point Software Emulation Package

Zilog

Product Specification

September 1983

FEATURES

- Provides high-quality floating-point arithmetic capability for Z8000 series CPUs.
- Executes the same instruction set and simulates the architecture of Zilog's Z8070 Arithmetic Processing Unit (APU). The same application software can use either this emulation package or the Z8070 APU without modification.
- Conforms to the proposed IEEE Standard P754 Draft 9.0 for binary floating-point arithmetic.
- Provides routines for the conversion of binary integer and Binary Coded Decimal (BCD) to and from binary floating-point formats.

GENERAL DESCRIPTION

The Floating-Point Software Emulation Package (also referred to as the Emulator) provides floating-point arithmetic capability in accordance with the proposed IEEE Standard for binary floating-point arithmetic.

The proposed standard is designed to facilitate the portability and increase the precision and reliability of numerically oriented programs. The Emulator's ability to handle various numeric formats makes it amenable to commercial as well as scientific applications. Because the Emulator simulates the Z8070 APU, it is an ideal tool for systems that may later include the Z8070 chip.

The Emulator is written in PLZ/ASM structured assembly language and is fully operable on the Z8000 series CPUs. It can be run in either segmented or nonsegmented modes and in either Normal or System modes. It is available with Zilog's Real-Time Software (ZRTS) and Zilog's System 8000.

The Emulator uses the Z8000 CPU's Extended Processing Architecture (EPA). The EPA function provides the capability for flexible hardware and software expansion by the addition of Extended Processing Units or software trap handlers. When the EPA bit in the CPU's Flag and Control Word is zero, the CPU traps to the Emulator upon encountering a floating-point instruction.

FUNCTIONAL DESCRIPTION

The Emulator consists of a small system-dependent module involved with memory accesses, and a system-independent body. The system-dependent module comes in PLZ/ASM source form, whereas the body is in object code. Two versions of the body are provided: one assembled for segmented operation, the other for nonsegmented use. The Emulator can, consequently, be run in either segmented or nonsegmented mode. The Emulator uses no privileged CPU operations, and most operations finish in under one millisecond on a 6 MHz CPU, including the trapping and typical operating system overhead.

The system-independent body is called by the system-dependent module, using Zilog's calling convention. The system-independent body consists of approximately 5000 bytes of code and requires fewer than 30 words of stack space for operation. After an operation, the body returns a status code, indicating whether or not a trap should be taken.

In the system-dependent module, the most important aspect of the Emulator is its method of accessing the address spaces of the process that generated the EPA trap. A small set of assembly language interface routines that can be tailored to the host system is provided in PLZ/ASM source form. These routines are fully general in that they permit trapped processes from either nonsegmented or segmented operating modes and from either System or Normal operating modes. Host systems not supporting some of these combinations can simply ignore them.

A user with an average mathematical background should be able to attain precise results with the floating-point arithmetic supported by the Emulator, because the software simplifies the development of accurate programs. The Emulator accepts numbers in any of several data formats, operates on them in the very precise Extended format, returns results to any of the formats, and indicates any exceptions that may arise.

Exceptions, as defined by the proposed IEEE Standard, include detection of invalid operands or results, attempted division by zero, and overflows or underflows caused by exceeding the limits of the data format. Exceptions can be handled by user-controlled traps, or can be dealt with in an automatic manner by the Emulator. In addition, the Emulator records historical information on exceptions for later evaluation. This creates a flexible environment in which the user can tailor exception handling based on the needs of a specific application.

The capabilities of the Emulator are of use wherever consistency of results, precision of results, manipulation of a wide range of numbers, or generally increased arithmetic processing capabilities are required. Such widely divergent applications as guidance systems, financial data processing, process control, graphics and robotics can be enhanced by increased arithmetic capabilities.

ARCHITECTURE

There are eight 80-bit data registers, two 32-bit status registers, one 16-bit and one 32-bit control register, and two 80-bit floating operand registers in the Emulator. This software architecture is designed to simulate the Z807C APU.

The Emulator has a data register file of eight 80-bit registers labelled F0 to F7. This format corresponds to the Double Extended format in which the Emulator performs all of its internal numeric operations. Figure 1 illustrates the data register file.

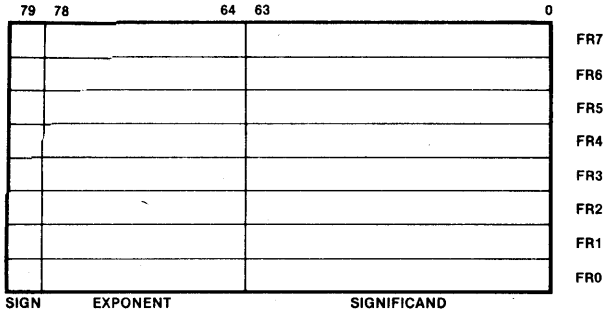


Figure 1. Data Register File

There are two 32-bit status registers known as the Program Counter register (PC1) and the Flags register. PC1 (Figure 2) holds the address of the instruction that generated the trap. The Flags register contains historical information on detected exceptions (sticky flags) and the Emulator's Compare and Remainder flags (Figure 3).

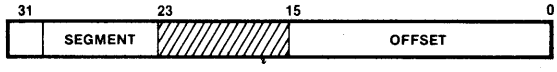


Figure 2. Program Counter Register (PC1)

Flags Register. Four fields provide information on exceptions and results. These fields are:

Sticky flags. Eight flags are set when an exception occurs, and remain set until they are cleared by the programmer. These flags are:

INV—Invalid
FOV—Floating-point Overflow
UN—Underflow
DZ—Divide-by-Zero
INX—Inexact Result
DE—Denormalized number
NAN—Signalling NaN
IX—Integer Overflow

Previous Operation flags. These are the same as the sticky flags, except these bits reflect the exception of the previous operation.

Compare and Remainder flags. These status flags correspond to the H, D, P/V, S, Z, and C flags in the Z8000 CPU's FCW.

FOP1E. These bits are the two most significant bits (MSB) of the exponent of Floating Operand register 1 (FOP1).

FOP2E. These bits are the two MSBs of the exponent of Floating Operand register 2 (FOP2).

There are two control registers in the Emulator: the System Configuration register and the User Control register. The System Configuration register is accessible only to privileged users in System mode; it contains interrupt controls and EPU information (Figure 4). The User Control register (Figure 5) is accessible to Normal mode users; it sets arithmetic modes and enables traps.

System Configuration Register. Ten fields provide information on instructions and interrupts. These fields are:

Interrupt Vector Number. Identifies the source and cause of an interrupt.

SV. Shifts the Interrupt Vector left.

VIS. Is set to include status information in the Interrupt Vector.

NV. Is set when no vector is to be returned.

IUS. Is set when an interrupt is under service.

Interrupt Pending (IP). Indicates a pending interrupt.

Master Interrupt Enable (MIE). Enables all interrupts.

Used (U). Indicates that a floating-point instruction has been executed.

ID. Indicates the number in the ID field of EPU instructions to which the Emulator will respond.

Invalid ID. Specifies which ID fields in EPA instructions should cause a trap to be generated.

User Control Register. Three fields enable traps and determine arithmetic modes. These fields are:

Rounding mode (RM).
00 = Round to Nearest
01 = Round toward Zero
10 = Round toward Plus Infinity
11 = Round toward Minus Infinity

Trap Enables. The setting of these bits enables the trap associated with each exception listed below.

INV—Invalid
FOV—Floating Point Overflow
UN—Underflow
DZ—Divide-by-Zero
INX—Inexact Result
DE—Denormalized number
NAN—Signalling NaN
IX—Integer Overflow

In addition, the Emulator contains two 80-bit floating operand registers labelled FOP1 and FOP2, which contain the input operand (FOP1) and result (FOP2) for use by trap handlers (Figure 6).

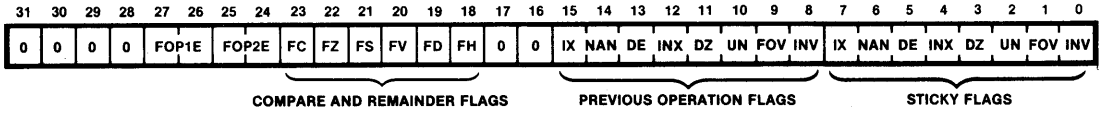


Figure 3. Flags Register

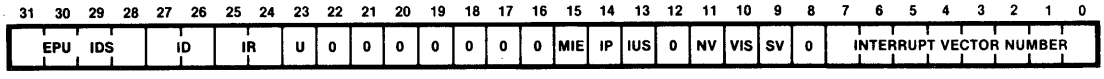


Figure 4. System Configuration Register

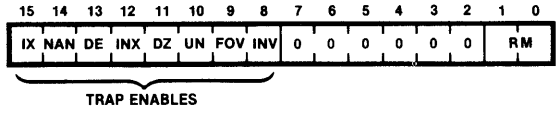


Figure 5. User Control Register

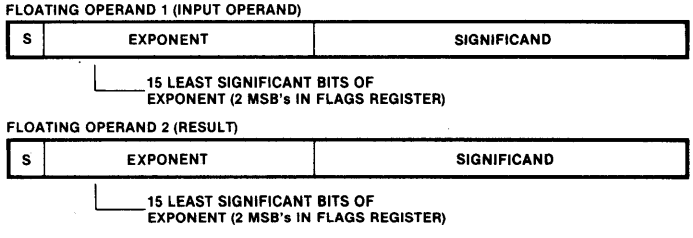


Figure 6. Floating Operand Registers

DATA TYPES

The Emulator supports the data types shown in Figure 7. All formats are automatically converted to the 80-bit floating-point format for internal operations and can be returned to any of the formats. The S bit is the Sign bit specifying a positive (0) or negative (1) number. The negative or positive floating-point number is equal to:

$$\text{significand} \times 2^{(\text{exponent-bias})}$$

The Emulator also supports extensions to the floating-point arithmetic. Infinities are represented, and numbers that cannot be represented in normalized form (i.e., where the most significant bit of the significand is a binary 1) can be represented in a denormalized form (i.e., with leading Os in the significand). In addition, certain values for NaNs (Not-a-Number) are defined, which are useful in causing traps or providing diagnostic information.

In the following description of the binary floating-point formats, 's' is the sign, 'e' is the exponent, 'f' is the significand (or fraction), and j is a 1-bit integer part. The integer bit is implicit in single and double formats.

The value (v) of the 32-bit Single Precision Binary format is determined as follows:

- If e = 255 and f ≠ 0, then v = NaN.
- If e = 255 and f = 0, then v = $(-1)^s$(infinity).
- If 0 < e < 255, then v = $(-1)^s 2^{e-127}(1.f)$.
- If e = 0 and f ≠ 0, then v = $(-1)^s 2^{-126}(0.f)$.
- If e = 0 and f = 0, then v = $(-1)^s 0$,(zero).

The value of the 64-bit Double Precision Binary format is determined as follows:

- If e = 2047 and f ≠ 0, then v = NaN.
- If e = 2047 and f = 0, then v = $(-1)^s$(infinity).
- If 0 < e < 2047, then v = $(-1)^s 2^{e-1023}(1.f)$.
- If e = 0 and f ≠ 0, then v = $(-1)^s 2^{-1022}(0.f)$.
- If e = 0 and f = 0, then v = $(-1)^s 0$,(zero).

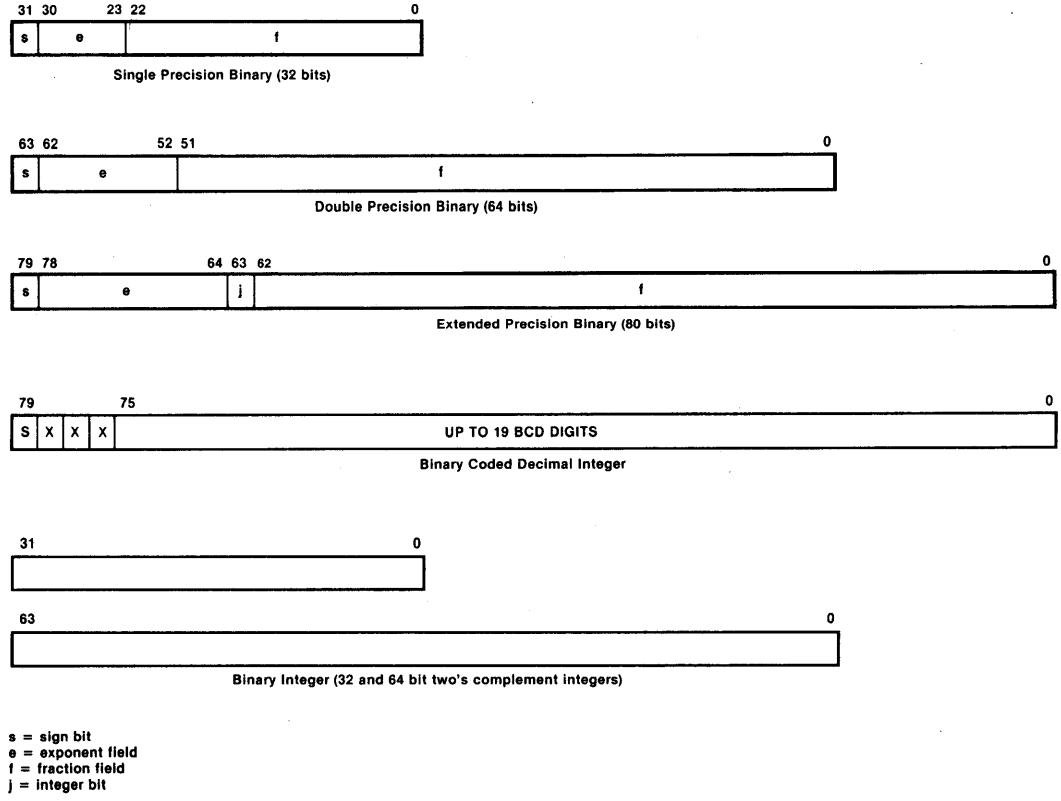


Figure 7. Data Formats

For the 80-bit Double Extended Precision Binary format, the value is determined as follows:

- If e = 32767 and f ≠ 0, then v = NaN.
- If e = 32767 and f = 0, then v = $(-1)^s$(infinity).
- If 0 < e < 32767, then v = $(-1)^s 2^{(e-16383)}(j.f)$.
- If e = 0 and j = f = 0, then v = $(-1)^s 0$, (normal zero).
- If e = 0 and j or f is nonzero, then v = $(-1)^s 2^{-16383}(j.f)$.

Floating-point instructions are of the form:

FXXX [S,D,L,Q,BCD] rnd dst,src

The suffixes for floating-point binary formats [Single (S), Double (D), or Extended (no suffix)], for binary integer formats [Long-word (L) and Quad-word (Q)], and for decimal format (BCD), refer to the size of the CPU register or memory operand. The suffix "rnd" refers to the rounding precision, or the degree of precision used in internal Emulator operations; SGL is used for Single precision, DBL for Double precision, and no suffix for Extended precision.

Addressing modes used include Emulator register (F) and multiple register (FF), CPU register (R), Index (X), Indirect Register (IR), Direct Address (DA). Also permitted are addressing of Emulator Control registers (FCTL) and portions of the Control registers (FSEL, traplist, flaglist, modelist).

INSTRUCTION SET

The floating-point instruction set provides the following types of instructions:

- Primary arithmetic operations
- Load and store operations
- Compare and examine operations
- Secondary arithmetic operations
- Control operations

Primary Arithmetic Operations

Mnemonic	Operands	Addressing Modes	Operation
FADD **FADDS** **FADDD**	dst,src	src: F,R,IR,DA,X dst: F	**Floating Add** dst ← dst + src
FDIV **FDIVS** **FDIVD**	dst,src	src: F,R,IR,DA,X dst: F	**Floating Divide** dst ← dst/src
FMUL **FMULS** **FMULD**	dst,src	src: F,R,IR,DA,X dst: F	**Floating Multiply** dst ← dst*src
FREMSTEP	dst,src	src: F dst: F	**Floating Remainder Step** dst ← dst src FCW ← flag
FSQR **FSQRS** **FSQRD**	dst,src	src: F,R,IR,DA,X dst: F	**Floating Square Root** dst ← SQR (src)
FSUB **FSUBS** **FSUBD**	dst,src	src: F,R,IR,DA,X dst: F	**Floating Subtract** dst ← dst − src
FLD **FLDS** **FLDD**	dst,src	src: F,R,IR,DA,X dst: F or src: F dst: R,IR,DA,X	**Floating Load** dst ← src

Load and Store Operations

Mnemonic	Operands	Addressing Modes	Operation
FLDBCD	dst,src	dst: F src: R,IR,DA,X or dst: R,IR,DA,X src: F	**Floating Load BCD Integer** dst ← Float (BCD_src) dst ← BCD (float_src)
FLDIL	dst,src	dst: F src: RR,IR,DA,X or dst: RR,IR,DA,X src: F	**Floating Load Binary Integer Long Word** dst ← Float (src) dst ← Fix (src)
FLDIQ	dst,src	dst: F src: RQ,IR,DA,X or dst: RQ,IR,DA,X src: F	**Floating Load Binary Integer Quad Word** dst ← Float (src) dst ← Fix (src)
FLDM	dst,src,n	dst: F,FF src: R,IR,DA,X or dst: R,IR,DA,X src: F,FF	**Floating Load Multiply** dst ← src
FLDTL	dst,src	dst: RR,IR,DA,X src: F	**Floating Load and Truncate to Integer Long Word** dst ← Int (src)
FLDTQ	dst,src	dst: RQ,IR,DA,X src: F	**Floating Load and Truncate to Integer Quad Word** dst ← Int (src)

Compare Operations

FCP **FCPS** **FCPD**	dst,src	dst: F src: F,R,IR,DA,X	**Floating Compare** dst − src, set flags
FCPF	dst,src	dst: F src: F	**Floating Compare and Transfer Flags to FCW** dst − src FCW ← flags
FCPFX	dst,src	dst: F src: F	**Floating Compare, Transfer Flags to FCW, and Raise Exception if Unordered** dst − src FCW ← flags
FCPX **FCPXS** **FCPXD**	dst,src	dst: F src: F,R,IR,DA,X	**Floating Compare and Raise Exception if Unordered** dst − src, set flags

Compare Operations (Continued)

Mnemonic	Operands	Addressing Modes	Operation
FCPZ	dst	dst: F	**Floating Compare with 0, and Transfer Flags to FCW** dst – 0 FCW ← flags
FCPZX	dst	dst: F	**Floating Compare with 0, Transfer Flags to FCW, and Raise Exception if Unordered** dst – 0 FCW ← flags

Secondary Arithmetic Operations

FABS **FABSS** **FABSD**	dst,src	dst: F src: F,R,IR,DA,X	**Floating Absolute Value** dst ← |src|
FCLR	dst	dst: F	**Floating Clear** dst ← +0
FINT **FINTS** **FINTD**	dst,src	dst: F src: F,R,IR,DA,X	**Floating Round to Floating Integer** dst ← Float [Int (src)]
FNEG **FNEGS** **FNEGD**	dst,src	dst: F src: F,R,IR,DA,X	**Floating Negation** dst ← (–src)

Control Operations

FLDCTL	dst,src	dst: FCTL src: RR,IR,DA,X or dst: RR,IR,DA,X src: FCTL	**Floating Load Control** dst ← src
FLDCTLB	dst	dst: Fsel	**Floating Load Control Byte** FCW ← flags
FRESFLG	src	dst: FFLAGS src: flaglist	**Floating Reset Flag** FFLAGS (flaglist) ← 0
FRESTRAP	src	dst: USER src: traplist	**Floating Reset Trap** USER (traplist) ← 0
FSETFLG	src	dst: FFLAGS src: flaglist	**Floating Set Flag** FFLAGS (flaglist) ← 1
FSETMODE	src	dst: FMODE src: modelist	**Floating Set Mode** FMODE ← modelist
FSETTRAP	src	dst: USER src: traplist	**Floating Set Trap** USER (traplist) ← 1

Condition Codes

A set of condition code mnemonics are provided for evaluating the results of floating-point comparisons when the results have been transferred to the CPU's Flag and Control Word. Table 1 shows the floating-point mnemonics, the equivalent Z8000 condition code mnemonics, and their meanings.

Table 1. Condition Code Equivalences

Floating-point CC	Z8000 CC	Meaning
FEQ	EQ	Equal
FNEU	NE	Not equal or unordered
FLT	ULT	Less than
FLE	ULE	Less than or equal
FGT	GT	Greater than
FGE	GE	Greater than or equal
FLU	LT	Less than or unordered
FLEU	LE	Less, equal or unordered
FGU	UGT	Greater than or unordered
FGEU	UGE	Greater, equal or unordered
FORD	NOV	Ordered
FUN	OV	Unordered

Programming Example

An example of a FORTRAN program and its possible compilation is provided below. The example calculates an average, using floating-point instructions and Emulator registers as well as CPU instructions and resources.

The FORTRAN segment assumes type REAL is a single precision number. The possible compilation of the FORTRAN segment assumes that the compiler optimizes variable usage in DO loops—it does not emulate full FORTRAN DO loop conditions. PLZ/ASM is the code produced.

FORTRAN Program Segment

```
      REAL SAMPLE(100),AVERAGE
      INTEGER INDEX
       •
       •
       •
      AVERAGE = 0.0
      DO 100 INDEX = 1,100
100   AVERAGE = AVERAGE + SAMPLE(INDEX)
      AVERAGE = AVERAGE/100.0
```

Possible Assembler Compilation

```
      SAMPLE ARRAY [100 LONG]
      AVERAGE LONG
      K100   LONG = 100.0
       •
       •
       •
                    !REGISTER ASSIGNMENTS!
                    !INDEX - R2!
                    !AVERAGE - F0!
      FCLR    F0
      CLR     R2
LOOP  FADDS   F0,SAMPLE(R2)
      INC     R2,#4
      CP      R2,#400
      JR      LT,LOOP
      FDIVS   F0,K100
      FLDS    AVERAGE,F0
```

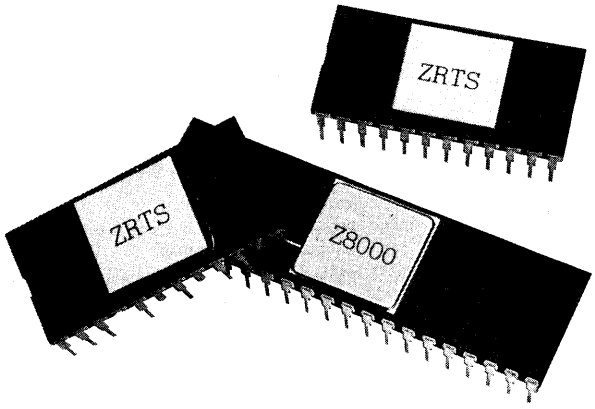
ZRTS™8000
Zilog Real-Time Software for the Z8000 Microprocessor

Zilog

Product Description

Preliminary

September 1983

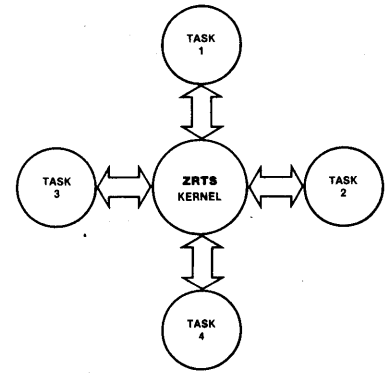


- **Real-time Multi-Tasking Software Components**
  - Synchronization of multiple tasks
  - Interrupt-driven priority scheduling
  - Real-time response
  - Dynamic memory allocation
- **Modular and Flexible Design**
  - Efficient memory utilization
  - 6K byte PROMable kernel
  - Support for Z8001 and Z8002 16-bit microprocessors
  - Configurable via linkable modules
- **Versatile Base for Z8000™ System Designs**
  - Segmented/non-segmented tasks
  - System/normal mode tasks
  - Uses standard Zilog calling conventions
- **Easy-To-Use System Generator**
  - High-level configuration language
  - Supports a wide variety of hardware configurations
  - Easily changed control parameters allow system optimization
  - Eliminates the requirement for intimate knowledge of system internal structure

OVERVIEW

Zilog's Real Time Software (ZRTS) provides of a set of modular software components that allows quick and easy implementation of customized operating systems for all members of the Z8000 16-bit microprocessor family. In effect, ZRTS extends the instruction set of the Z8000, adding easy-to-use commands that give the Z8000 the capability for managing real-time, multi-tasking applications.

The ZRTS package consists of a small real-time, multi-tasking executive program, the Kernel, and a System Configurator. The Kernel provides sychronization and control of multiple events occurring in a real-time environment. All major real-time functions are available—task synchronization, interrupt-driven priority scheduling, intertask communication, real-time response, and dynamic memory allocation. The System Configurator is a language processor that allows the target operating system to be defined in high-level terms using the ZRTS Configuration Language (ZCL).



These functions greatly simplify the tasks of the designer, allowing development efforts to be concentrated on the application, instead of on real-time coordination, task management problems, and complicated system generations. ZRTS provides a modular and flexible development tool that serves as a versatile base for Z8000 system designs. The Kernel requires only 6K bytes of either PROM or RAM memory, thus allowing configurations for a wide variety of target systems, while producing a memory-efficient cost-effective end product

CONCEPTS

ZRTS is both easy-to-learn and easy-to-use. Only a few simple concepts need to be understood before designing begins.

Tasks. Tasks are the components comprising a real-time application. Each task is an independent program that shares the processor with the other tasks in the system. Tasks provide a mechanism that allows a complicated application to be subdivided into several independent, understandable, and manageable units.

Semaphores. Semaphores provide a low overhead facility for allowing one task to signal another. Semaphores can be used for indicating the availability of a shared resource, timing pulses or event notification.

Exchanges and Messages. Exchanges and Messages provide the mechanism for one task to send data to another. A Message is a buffer of data, while an Exchange serves as a mailbox at which tasks can wait for Messages and to which Messages are sent and held.

Two components make up the heart of ZRTS—the ZRTS Kernel and the ZRTS Configuration Language (ZCL). An I/O subsystem and a debugger are also provided in ZRTS.

The ZRTS Kernel. The Kernel is the basic building block of ZRTS and performs the management functions for tasks, semaphores, the real-time clock, memory and interrupts. The Kernel also provides for task-to-task communications via Exchanges and Messages. All requests for Kernel operations are made via system call instructions with parameters in registers, according to the standard Zilog calling conventions.

Task Management. One of the main activities of the Kernel is to arbitrate the competition that results when several tasks each want to use the processor. Each task has a unique task descriptor that is managed by the Kernel. The data contained in the descriptor includes priority, status and other pertinent information. ZRTS supports any number of tasks, limited only by the memory available to accommodate the task descriptors and stacks.

The Kernel maintains a queue of all active tasks on the system. Each task is scheduled for processor time based on its priority. The highest-priority task that's ready to run gains control of the CPU; other tasks are queued. Tasks can be prioritized up to 32767 levels, with round-robin scheduling among tasks with the same priority.

Tasks can run either segmented or non-segmented code, in either normal

TABLE 1.

TASK MANAGEMENT	
T_Census	Provides the status of tasks in the system.
T_Create	Creates a task dynamically.
T_Destroy	Removes a dynamically created task.
T_Lock	Allows a task to take exclusive control of the CPU.
T_Reschedule	Changes the priority of a task.
T_Resume	Activates a suspended task.
T_Suspend	Suspends another task.
T_Unlock	Releases exclusive control of the CPU for other tasks.
T_Wait	Suspends task execution.
T_Whoami	Returns the name (address) of the task making the request.
SEMAPHORE MANAGEMENT	
Sem_Clear	Clears semaphore queue and reinitializes a semaphore.
Sem_Create	Creates a semaphore dynamically.
Sem_Destroy	Removes a dynamically created semaphore.
Sem_Signal	Signals a semaphore, increments the counter.
Sem_Test	Tests a semaphore for a signal.
Sem_Wait	Causes a task to wait until a semaphore is signaled, decrements the counter.
CLOCK MANAGEMENT	
Clk_Delay_Absolute	Places a task on the clock queue waiting for absolute time.
Clk_Delay_Interval	Places a task on the clock queue waiting for passage of an interval of time.
Clk_Set	Sets the real-time clock.
Clk_Time	Reads the clock.
MEMORY MANAGEMENT	
Mem_Census	Provides status of the memory resource.
Alloc	Dynamically allocates memory.
Release	Releases allocated memory.
INTER-TASK COMMUNICATION	
M_Acquire	Gets a message from an exchange pool and assigns a destination or a reply exchange to it.
M_Assign	Assigns a new source and destination to an existing message.
M_Create	Creates a message dynamically.
M_Destroy	Removes a dynamically created message.
M_Get_Descriptor	Gets message's descriptor information
M_Read	Reads the message data.
M_Receive	Receives a message from an exchange.
M_Receive_Wait	Waits to receive a message from an exchange.
M_Release	Returns a message to the exchange pool.
M_Reply	Sends a message back to destination exchange.
M_Send	Sends a message to an exchange.
M_Write	Changes message data.
X_Create	Dynamically creates an exchange with a pool of messages.
X_Destroy	Removes a dynamically created exchange.

or system mode. The numerous operations that may be performed on tasks are listed in *Table 1.*

Semaphore Management. The Kernel provides semaphore management for synchronizing interacting tasks. A typical use of semaphores is to provide mutual exclusion of a shared resource. When a resource is to be used by only one task at a time, a semaphore with a counter of 1 controls the resource. Every task requiring the resource must first wait on that semaphore. Since the counter is 1, only one task will acquire the resource. The others will be queued on the semaphore and suspended until the semaphore is signaled that the resource is once again available. At that time, the first task on the semaphore queue will be made ready to run and can use the resource. After all tasks have acquired the resource and signaled the completion of their use, the semaphore returns to its original state with a counter of 1. Counters greater than one are useful when there are a number of similar resources, (i.e., three tape drives, four I/O buffers, etc.).

In ZRTS, a semaphore can count up to 32767 signals. The commands provided by the Kernel to manage semaphores are listed in *Table 1.*

Clock Management. ZRTS operates with a real-time clock that generates interrupts at a hardware-dependent rate. It is used for timed waits, timeouts, and round-robin scheduling. All times are given in number of ticks. The clock may be manipulated by the set of commands provided by the Kernel that are listed in *Table 1.*

Memory Management. Storage for ZRTS data structures is allocated either statically at system generation time, or dynamically at run time. Dynamic allocation occurs via a system call that specifies the attributes of the structure to be created and returns a name that can be used to refer to the structure. Memory is allocated in 256-byte increments, and can be released using a system call.

The storage allocator can also be called directly to obtain blocks of memory up to 64K bytes long, which can be used by the task for any purpose.

Interrupt Management. Interrupt-handling routines are provided for system calls, non-vectored interrupts and a hardware clock. The user must provide interrupt routines for whatever other vectored interrupts are included in the target system.

ZRTS can switch control to a task waiting for an external event within 600-microseconds after the occurrence of the event. This is a worst-case time for a system using a 4 MHz Z8001 CPU and is based on a Sem_Signal system call awakening a higher priority task that is waiting on a semaphore; this causes a task switch to occur.

TABLE 2.

CONSTANTS	Specifies system constants.
EXCHANGES	Defines the characteristics of application exchanges.
INITIALIZATION	Specifies routines that are to execute prior to beginning execution of the first task.
INTERRUPT	Associates an interrupt routine with an interrupt vector or trap and system call-handlers. Provides the facilities to specify a NVI interrupt-handler that will be called from the system NVI-handler routine.
MEMORY	Defines sections or segments that contain code, initialized data, or uninitialized data and specifies the location in memory where it will be placed. The files to be included in the configuration are also defined in this section in conjunction with the section/segment definitions.
SEMAPHORES	Defines the characteristics of application semaphores.
SWITCHES	Allows flags that control the system generation operation to be set.
TASKS	Defines the characteristics of application tasks.

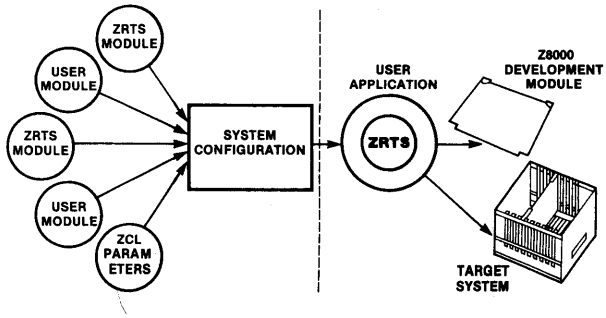
Inter-Task Communication. The Kernel provides the capability for tasks to exchange information. This communication process occurs when one task sends a Message to an Exchange and another task receives the Message.

A Message contains a length indicator, a buffer with a variable amount of data, and a code that identifies the Message type. The Exchange is a system data structure that consists of a queue for Messages sent but not yet received, a semaphore on which a task can wait for a Message, and an optional "pool" list from which Messages can be obtained quickly.

ZRTS provides several commands for inter-task communications. These are listed in *Table 1.*

Logical I/O. ZRTS includes an optional module which provides a device-independent mechanism for interfacing between tasks and customer-written I/O device drivers. Sample device drivers are included for terminal and disk-type devices.

ZRTS Configuration Language (ZCL). Since ZRTS's modular design leads to so many different configurations, a simple facility for generating the target operating system is a critical part of the ZRTS package. The ZRTS Configuration Language (ZCL) provides an easy-to-use means for generating the target system. Using ZCL, the designer can specify hardware information, software parameters, linkage information, and system data structures in high-level terms.



Development Environment

ZCL unburdens the user of the necessity to learn the details of the ZRTS internal structures. System data structures can be generated simply by specifying the appropriate parameters. The ZCL syntax is free-format with comments allowed to make the configuration commands more readable and maintainable.

ZCL input is comprised of a number of descriptive sections, each containing the details of the target operating system. The functions of these sections are described in *Table 2.* A sample system generation using ZCL is illustrated in *Figure 1.*

DEVELOPMENT ENVIRONMENT. Application modules for ZRTS are developed on a Zilog System 8000. The application or system generated can then be downloaded into a Zilog S8000 Development Module or a customized target system.

An interface package is provided for making ZRTS system calls from programs written in C, Pascal, or FORTRAN. Register usage by the system calls is compatible with Zilog's calling conventions.

A debugger is provided with ZRTS for use in testing and debugging applications. After the application is debugged, the system can be easily reconfigured (without the debugger) into its final form.

```
SWITCHES:

    application  segmented

INTERRUPTS:

CONSTANTS:

MEMORY:

    section     = [ type = code, base = <<0>>%7800 ]
    section     = [ type = data, base = <<0>>%8000 ]
    free_memory = [<<0>>%c000..%ffff]

    files =     timer.Z          ! configuration !
                timer.OC.I       ! demo code     !
                timer.IO.I       ! IO & support  !
                polled_sio.I     ! display I/O   !
                demio.I          ! display primitives!

INITIALIZATION:

    SIO_Init

TASKS:

    input_handler_task
        = [ entry = INPUT_HANDLER, priority = 10, status = ready ]

    time_display_task
        = [ entry = TIME_DISPLAY, priority = 20; status = ready ]

    egg_timer_task
        = [ entry = INTERVAL_TIMER, priority = 20, status = ready ]

    alarm_task
        = [ entry = ALARM, priority = 20, status = ready ]

    one_second_task
        = [ entry = ONE_SECOND_GENERATOR, priority = 30, status = ready ]

SEMAPHORES  :

        ONE_SECOND_SEMAPHORE
        TIME_DISPLAY_ENABLE_SEMAPHORE

EXCHANGES   :

        INPUT_HANDLER_ETR_EXCHANGE      = [ messages = 1, message_size = 8 ]
        INTERVAL_TIMER_ENABLE_EXCHANGE  = [ messages = 0 ]
        INPUT_HANDLER_A_EXCHANGE        = [ messages = 1, message_size = 8 ]
        ALARM_EXCHANGE                  = [ messages = 1, message_size = 8 ]
```

Figure 1. ZCL Sample Input.

ORDERING INFORMATION

Description	Prerequsites
ZRTS Zilog Real-Time Software for the Z8000	System 8000 (Requires Software License)

Z80® PLZ

Zilog

Product Description

June 1982

- **High-Level Procedure-Oriented Language Permits Efficient Writing of Machine-Independent Modules and Programs.**
- **Structured Format for Fast and Easy-to-Compile Programs.**
- **Produces Efficient Code for Economical Memory Usage and Processing Time.**
- **Simplifies Software Production and Maintenance.**
- **Allows Direct or Interpretive Execution of Program Modules.**

OVERVIEW

Z80 PLZ is a family of different programming languages designed to satisfy a wide range of microcomputer software development requirements. The two members of the PLZ family, PLZ/SYS and PLZ/ASM, produce object code-compatible modules and share common control structures and data definition facilities. Thus, selective portions of programs may be written in the most appropriate language for the specific application and still maintain a consistent structure between modules.

PLZ/SYS is a high-level, procedure-oriented language that is syntactically similar to PASCAL. It provides a medium for writing structured, machine-independent programs with a minimum of programming effort.

PLZ/ASM, on the other hand, is a structured assembly language that permits access to the low-level capabilities of the processor by mixing assembly language and high-level control structures.

FEATURES

Compiler. The Z80 PLZ/SYS Compiler translates source code modules into an intermediate stage called Z-code. The Z-code modules may then be executed interpretively or processed by the code generator to produce a machine-code object module.

Code Generator. The Z80 PLZCG Code Generator accepts a file of intermediate Z-code generated by PLZ/SYS and produces the corresponding Z80 machine code as a relocatable object module. This file may be linked with other modules to form the complete executable load module.

Interpreter. The intermediate Z-code modules produced by the Z80 PLZ/SYS Compiler can be executed interpretively by ZINTERP. Linking ZINTERP with the other modules generated by the compiler produces an executable load module.

PLZ/ASM Translator. The PLZ FILTER translates a PLZ/ASM source module into a file of the corresponding Z80 Assembler source. This gives the Assembler the benefit of logical data structure, program flow control, and modular program design, in addition to its existing features.

PLZ Linker. The PLZ Linker, PLINK, links Z-code, ZINTERP and/or machine code modules into a single relocatable load module, allowing the user to control the overall size and speed of the program.

Although interpretive Z-code runs more slowly than machine code, the space savings over machine code is usually substantial for larger programs where the 3K bytes of ZINTERP is a small percentage of the entire program. By balancing the number of Z-code and machine code modules, the user can maximize the efficiency of a particular program.

PLINK resolves any external references between separately assembled modules, so that the load module produced is relocatable. It also allows the reordering and combining of named sections between modules and supports incremental linking.

ORDERING INFORMATION

Part No.	Description
07-3301-01	Z80 PLZ Object Diskette for use with PDS 8000/05 and PDS 8000/15
07-3302-01	Z80 PLZ Object Diskette for use with ZDS-1 Series
07-3303-01	Z80 PLZ Object Cartridge Disk for use with PDS 8000/20 and PDS 8000/30

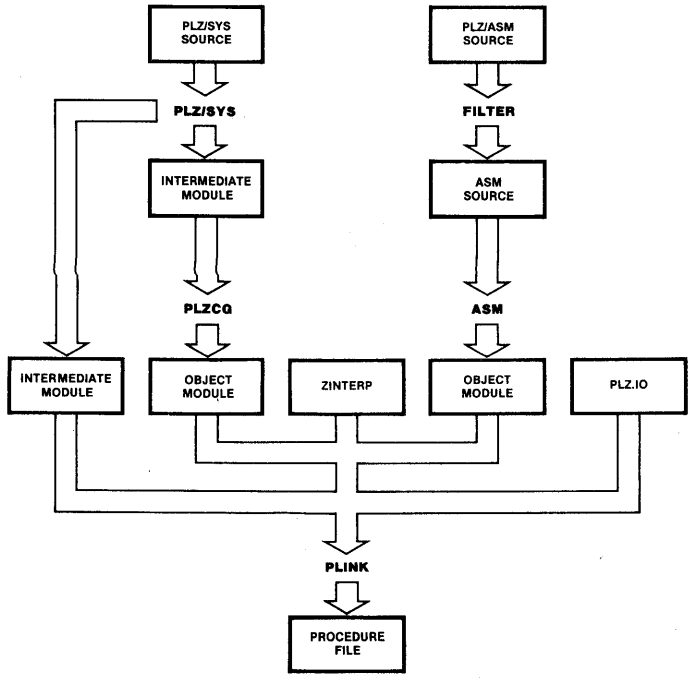


Figure 1. Z80 PLZ Language Modules.

Z800™ Cross Software Package

Zilog

NEW 1984

Product Description

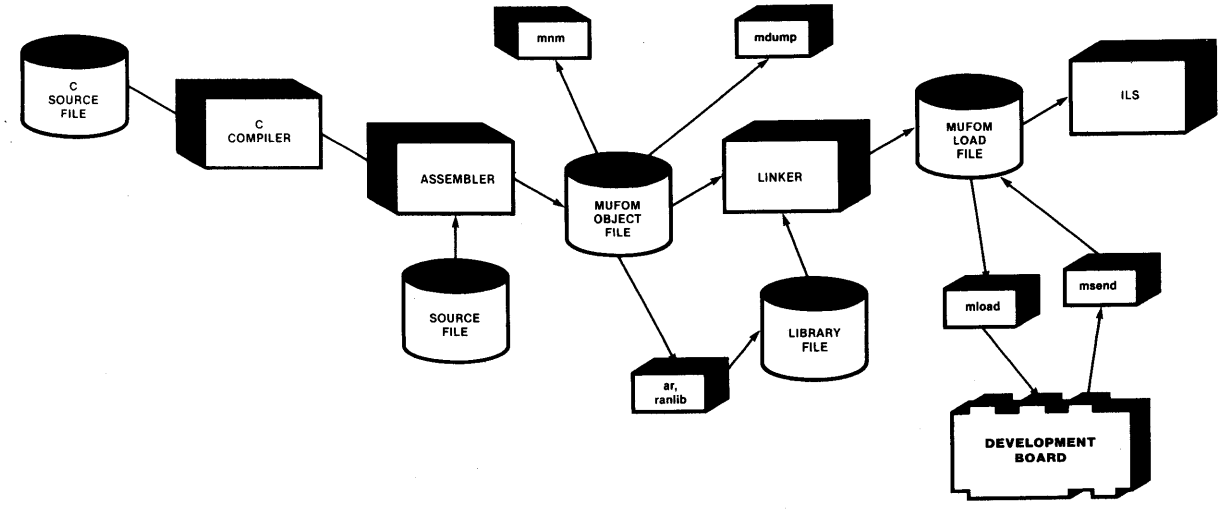
September 1983

OVERVIEW

The Z800 Cross-Software Package is a complete development environment that provides all the tools necessary to both generate and debug programs for the Z800 MPU. The package—a comprehensive collection of software routines and utilities—allows programmers to prepare, modify, test, and debug their software prior to release of the Z800 chip. The result of this comprehensive software testing and evaluation is a high-quality design that will be ready for end use by the time the hardware has been perfected.

The user interface for each of the program tools is similar to the standard UNIX* interface. Code testing and debugging can be done either with the Instruction Level Simulator (ILS) on the host system or by downloading to a Z800-based target system; mload and msend utilities are included to manage the host-to-target system communication. Use of the ILS facility allows program debugging well in advance of prototype hardware availability.

All the software tools in the Cross-Software Package are designed to run on either the DEC-VAX or the Zilog ZEUS S8000, both with a UNIX operating system. The programmer can choose either the C programming language or Z800 (Z80-compatible) assembly language to develop, load, test, debug and run programs for the Z800 MPU.



Z800 Cross-Software Package

*UNIX is a trademark of Bell Laboratories.

FEATURES

The Z800 Cross-Software Package consists of the following tools:

- **C Cross Compiler.** Produces efficient assembly-language code. Standard run-time library included.
- **Cross Assembler.** Relocatable assembler with macros, conditional assembly, and floating-point support.
- **Linker.** Handles full address range of the Z800 MPU using proposed IEEE-standard MUFOM format.
- **Instruction Level Simulator.** Debugging environment for Z800 code on cross-software host.
- **Object File Utilities.** Standard UNIX object utilities for MUFOM files, plus upload/download to target.
- **Portable Debugger.** Written in C language, the debugger facilitates the use of different hosts. It not only runs on Z800 target systems, but is also the user interface to the ILS.
- **Test cases.** Included with each tool for product testing.
- **Documentation** for each tool in the package.

DESCRIPTION

C Cross Compiler

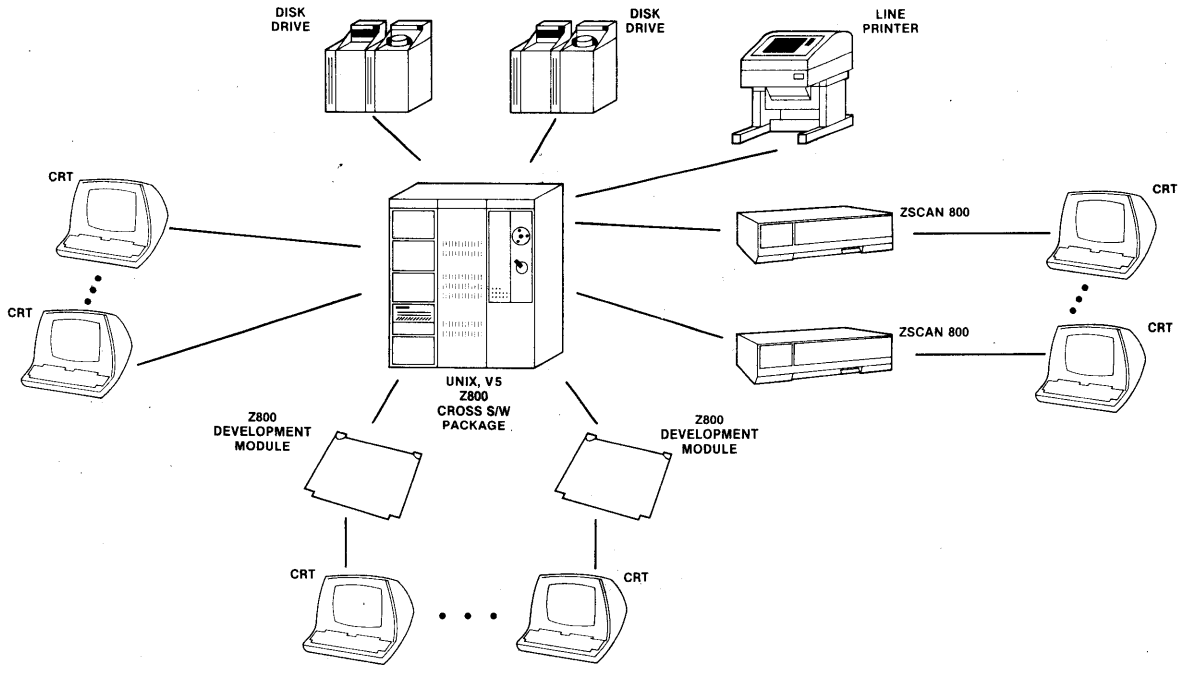
The Z800 C Cross Compiler is a UNIX-license-free C compiler, designed to run on either Zilog's UNIX S8000 or the DEC-VAX. The Z800 Compiler includes a Bell System V-compatible front end, and a back end generated by an automatic code-generator. Running at speeds similar to the Portable C Compiler, the Z800 Compiler features a variety of optimizations, some machine-independent, others machine-dependent.

The compiler supports the latest version of the C programming language, and thus includes longword and floating-point support, enumerations, structure assignment, and full "define" capability. The compiler can automatically invoke the C optimizer to produce more efficient code. The compiler produces assembly language code as output to allow easy debugging of the compiled code, or to allow hand optimization for critical code sections. A listing file can be generated that shows the assembly language interspersed with the C code that produced it.

Options include:

- **o** invokes the C optimizer.
- **s** compiles the source files but suppresses the assembly and link steps.
- **c** compiles and assembles the source files but suppresses linking.

Without the use of these options, the C Cross Compiler by default compiles, assembles, and links Z800 code.



Typical Z800 Cross-Software Package Installation

Cross Assembler

The Z800 Cross Assembler utilizes expanded Z80 mnemonics and addressing modes to assemble and generate MUFOM object modules (IEEE proposed standard #P695). The Z800 Cross Assembler is both fast—assembling at least 1,000 lines per minute—and efficient—consuming no more than 64K bytes of code/data space. Written in the C programming language, the assembler is designed to be easily retargetted. Additional features include macros, conditional assembly, and relocation. The Z800 assembler is upwardly compatible with Zilog's Z80 assembler and Microsoft's Macro-80 assembler.

The assembler supports a full set of pseudo-ops including conditional pseudo-ops and a powerful macro capability as well as pseudo-ops to support floating-point operations. A cross-reference listing can be generated to show the use and definition of all the symbols in the program.

The assembler supports the complete list of opcodes in the Z800 MPU Preliminary Product Specification (document # 00-2259-01), plus the instruction set of the Z8070 Floating-Point Unit. All Z80 opcodes, pseudo-ops, and commands supported by the Z80-RIO assembler are also supported. Constants supported by the assembler include integers, floating-point numbers, characters, and character strings. In addition, the assembler supports expressions using up to 80 bits of precision.

Addressing modes for the assembler are also derived from the Z80 assembler:

- Register
- Immediate
- Indirect Register
- Direct Address
- Indexed
- Short Index
- Based Indexed
- Stack Relative
- PC Relative

Three kinds of macros are included in the Z800 assembler:

MACROs are compatible with the Z80 assembler. Parameters are separated by blanks or commas and are substituted into the macro body as strings.

PROCs are call-by-value macros. Parameters are expressions, separated by blanks or commas, and are substituted into the macro body as values.

FUNCs are function macros. They are similar to PROC macros, except that they are recognized within expressions as operators or operands.

Additional pseudo-ops are provided to extend macros, conditional assembly, data definition, and object code generation. A command-line option specifies a third pass for generating a cross reference.

Linker

The Z800 linker is a processor-independent linker that combines MUFOM format-relocatable object modules generated by the cross assembler and resolves external references from within the modules or by searching object code libraries. The result is an executable program that can be sent to a target system for execution or simulated on the ILS. The linker also provides many advanced features such as complex expression evaluation with up to 80 bits of precision, multiple program/data segments of any length, and memory protection attributes for any portion of the memory space.

Object File Utilities

Z800 Cross Software Package includes a number of machine-independent utilities for processing object modules created in MUFOM format. Because these utilities can be used for different computer systems without modification, the programmer is freed from such constraints as variations in address and data size; these potential snags are resolved by the MUFOM utilities themselves.

MUFOM allows for up to 256 independently relocatable sections per module. This gives the programmer flexibility and control over how the program is mapped in memory. Program size is approximately 96K bytes, with 64K bytes for code and 32K for data.

Object modules can be stored in either ASCII or binary format. While ASCII is useful for downloading across serial links, binary modules require less file space and allow faster processing.

For linking and loading object modules, two utilities are provided: mlink and mload. mlink resolves external references among modules, relocates addresses, and combines the object modules into a single module called a load module. mload downloads an object file from the host system to the target system. mload and mlink can send or receive object files in either Tektronix hex format, Intel hex format, or ASCII MUFOM format.

In addition to mlink and mload, there are several other functions performed by the MUFOM utilities: mnm, msend, mdump, mcb, and mbc. These facilities allow inspection of object code files and the creation and maintenance of object libraries for use by the linker.

mnm prints the name list (symbol table) of an object module.

msend uploads the contents of memory from a target system and creates an absolute MUFOM object module on the host system.

mdump dumps a hex object file along with relocation information; object header information is also displayed.

mcb converts an ASCII object module into a binary module.

mbc converts a binary object module into ASCII.

Portable Debugger

The Z800 Portable Debugger is a versatile debugging system, able to run on either Z800 (8-bit) or Z8000 (16-bit) microprocessor systems. The debugger is written in C language, using the standard C library as well as a library for stand-alone I/O and memory utilities appropriate to the target system. To facilitate the use of different hosts (e.g., other UNIX systems, other operating systems such as CP/M, or development module environments), the processor descriptor parameters, such as state structure/display, memory and I/O access, Mini Assembly and Disassembly library routines (MAD) and the ILS, are contained in separate, easily modified modules.

Instruction Level Simulator (ILS)

The Z800 ILS is a powerful software tool that simulates execution of Z800 instructions. These instructions can then be displayed and manipulated by the portable debugger. The ILS facility executes Z800 instructions at slower than real-time rates and allows access to the simulated processor registers and to the full range of simulated memory. The ILS supports both symbolic and mnemonic Z800 opcode assembly, disassembly, and emulation. Additional features of the ILS include simulation of the on-chip MMU to provide a large (24M byte) addressing space; multiple breakpoints; single-stepping; I/O simulation; full memory, register and flag access; and friendly error reporting.

The ILS software program accepts as input Z800 machine code generated by the Z800 assembler; it then simulates changes in processor state. Each simulation is a single-step routine that works in conjunction with the debugger.

A typical process for using the ILS to generate, assemble, and debug Z800 software is as follows:

1. A conventional editor is used to create a Z800 assembly language source file.
2. The source file is then assembled and tested for syntax errors.
3. The default MUFOM object file produced by the assembler, named m.out, is then loaded into the debugger/ILS and run.

Z8® Software Development Package

Zilog

Product Description

September 1983

- **Structured Assembly Language with High-Level Constructs.**
- **Relocatable and Absolute Object Code Format.**
- **Free Format Statements Allow Indentation and Spacing for Readability.**
- **External Symbol References.**
- **Global Symbol Definitions.**

OVERVIEW

The Z8 Software Development Package consists of five utility programs which aid and simplify software development for Z8-based systems. Z8 PLZ/ASM, part of Zilog's PLZ family, brings all the advantages of modular programming to Z8 software development. The programming task can be broken into easily managed modules, giving more work assignment options to the engineering manager and a clear-cut structure to the individual programmer. The Z8 linker completes the task by combining the modules and resolving any external references.

FEATURES

Assembler. The Z8 PLZ/ASM Assembler translates easy-to-read, free-format PLZ/ASM source programs to object code. Because the user may specify that either absolute or relocatable object code be produced, he may choose a memory location for the program or leave that responsibility to the Linker. The Z8 PLZ/ASM Assembler produces a listing file containing both the source and assembled code.

Z8 PLZ/ASM allows an efficient mix of powerful assembly language mnemonics with high-level control structures such as IF . . . THEN . . . ELSE . . . FI and DO . . . OD loops. The PLZ/ASM programmer may map instructions and information into the Z8's register, program and data memory spaces, and organize the data space with such data declarations as RECORDS and ARRAYS. The PLZ/ASM Assembler supports external symbol references and global symbol definitions and is fully supported by the RIO™ operating system.

ZLINK. ZLINK links assembled modules into a single relocatable module and resolves any external references among separately assembled modules. It can also reorder and combine named sections found in the input assembly language modules. ZLINK accepts a symbolic specification of the program entry point in the command line and, on request, produces a detailed link map which gives the locations of global references and relocated modules and sections. Errors in the linking process are reported in the optional link map and at the system console.

Imager. IMAGER accepts multiple linked-object files from the linker and translates them into absolute code. IMAGER can then either store the absolute code in a disk file or leave it in system memory. Named sections found in the input object modules may be reordered and loaded anywhere in system memory.

Program Transfer. LOAD/SEND downloads an absolute program file into the Z8 Development Module for debugging, then sends it back to the disk for back-up and storage.

Prom Programming. Z-PROG stores the perfected load module in PROM.

ORDERING INFORMATION

Prerequisites:
PDS 8000 Series
ZDS 1/40 or 1/25
MCZ-1 Series
RIO

Part No.	Description
07-0086-01	Z8 Software Development Package Object Cartridge Disk for Use with PDS 8000/20A
07-3361-01	Z8 Software Development Package Object Diskette for Use with PDS 8000/5 and PDS 8000/15
07-3362-01	Z8 Software Development Package Object Diskette for Use with ZDS-1 Series
07-3363-01	Z8 Software Development Package Object Cartridge Disk for Use with PDS 8000/20 and PDS 8000/30